Orbital Viewer

by David Manthey

February 1998

Updated September 2004

TABLE OF CONTENTS

INTRODUCTION

Orbital Viewer is a program for visualizing atomic and molecular orbitals. Orbitals are the electron probability functions which are computed from Schrödinger's Equation. The example shown in Figure 1 is a 4f0 orbital ($n = 4$, $l = 3$ (f), $m_l = 0$), plotted with a surface of probability where $\Psi^2 = 10^{-4}$. It is illuminated with two light sources to better show its shape.

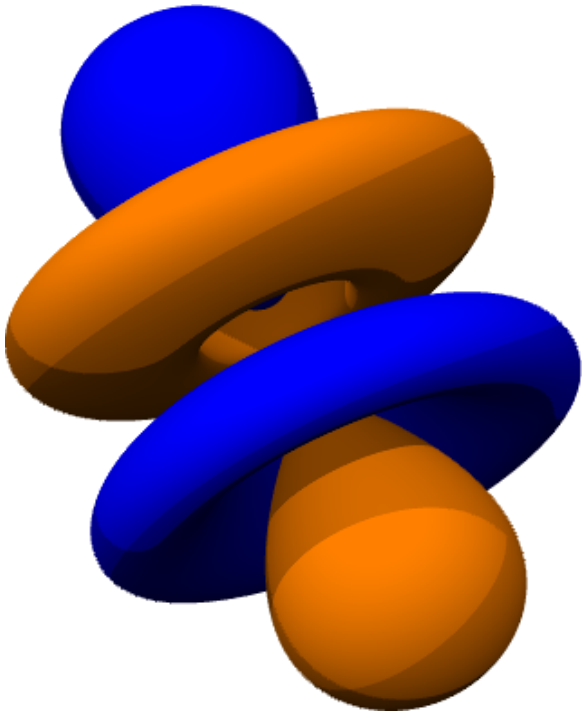


Figure 1: 4f0, $\Psi^2=10^{-4}$

There are two different versions of Orbital Viewer. The first is a graphics program, compatible with Windows 95, Windows NT, and Windows 3.x (with WIN32s installed). This program requires an 80386 processor or better, and is called OV.EXE. The second version is a command line program which can theoretically be run on any machine which has an ANSI C compatible compiler. This program is called ANSIORB.EXE (or something similar, as some computers do not require the .EXE extension to determine file type).

Although this manual is focused on OV.EXE (the graphics version), almost all of the options available through the graphics interface are available to the command line version. The principal difference between them is that the graphics version provides an easier way to manipulate and modify an orbital, while the command line version is typically faster and runs on more machines. Both versions can read the same orbital specification file format.

This program can display orbitals in three general ways:

(1) Probability density: This shows the probability at each point in the orbital based on how intense the colors are, or how many points are present. See Figure 2a.

(2) Surface of probability: This is a surface of constant probability. Like an line on a contour map, this displays the location where the probability has a constant value. See Figure 2b.

(3) Asymptotes: These are the conic sections, spheres, and other shapes where the probability is zero. See Figure 2c. The phase of the orbital changes at the asymptote.

Figure 2a: 4f0, probability density plot

Figure 2b: 4f0, surface of probability, $\Psi^2=10^{-4}$

Figure 2c: 4f0, asymptotes

Throughout this manual, orbitals have different colors for the positive and negative phases. Phases are taken as defined in Condon and Shortley (see References). In all figures the blue or darker color is the positive phase while the orange color or lighter color is the negative phase.

There are also three general types of display:

(1) Point density plot: This is a probability density representation of the orbital. Points are randomly selected based on the probability of the orbital. See Figure 3a. This can be combined with a polygon asymptote plot. None of the lighting options apply to this style.

(2) Polygon surface plot: This is an approximation of a surface of constant probability, with the vertices of the polygons at the surface. See Figure 3b. This can be combined with a polygon asymptote plot. Opacity is approximated, and shadows and indices of refraction can not be applied to this style.

(3) Raytracing: This style supports the most features, including surface, probability, and interior opacity, precise asymptotes, point light sources with shadows, and indices of refraction. See Figure 3c. This is also the most computationally intensive style.

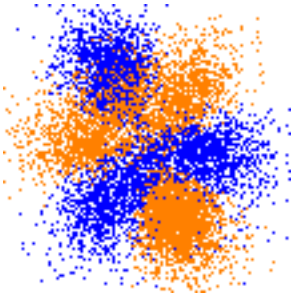


Figure 3a: 4f0, point density plot

Figure 3b: 4f0, polygon surface plot

Figure 3c: 4f0, raytraced surface plot

There are many options within each display type. Additionally, lighting can be adjusted, a cutaway can be added (to show the interior of the orbital), and six different forms of stereo displays are supported (to show the orbital in 3D). This manual includes brief examples of the different functions, plus a guide to the input file format, and a section on the equations used to calculate the orbitals.

ORBITALS

The most important item that can be specified in Orbital Viewer is the atom or molecule which will be drawn. This is specified in the **Display / Orbitals** menu option. Selecting this displays the dialog shown in Figure 4. Any positive number of atoms can be specified.

Figure 4: Orbital Dialog

Each atom is specified by three quantum numbers: n, l, and m. n is the principal quantum number, l is the orbital quantum number, and m is the angular momentum quantum number (sometimes referred to as the magnetic quantum number). m is often written as m_l, with the spin quantum number written as m_s. The spin quantum number does not effect the shape of the orbital, and, therefore, is not an adjustable parameter. n can be any positive integer. However, due to the limits of computer precision, n is restricted to the range of 1 to 30. l can be any value between 0 and n-1, inclusive. m can be any value between $-l$ and $+l$, inclusive. The actual shape of the orbital is dependant on n-l, l-$|m|$, and m.

The orbital quantum number, l, is often designated as a letter. Letter designations are given to l values from 0 to 20 as follows: {s, p, d, f, g, h, i, k, l, m, n, o, q, r, t, u, v, w, x, y, z}, where s corresponds to l=0, p to l=1, d to l=2, and so forth. The first four of these are due to spectroscopy historical reasons, with s being short for *sharp*, p for *principal*, d for *diffuse*, and f for *fine*. The remaining letters were assigned in order, skipping those which where thought to cause confusion.

All orbitals are computed using the hydrogenic equations, but any atom can be specified by giving the number of protons (sometimes referred to as Z), and the atomic weight. A reasonable approximation of atomic weight is (Z+N) amu, where N is the number of neutrons, and an amu (atomic mass unit) is $1.6605402 \times 10^{-27}$ kg. Increasing the mass or the number of protons has the effect of shrinking the radius of the orbital.

The factor is a multiplication factor for the atom's probability. Typically, it is either 1 or -1. -1 will reverse the phases of the orbital, which is useful in molecule construction. See the example on H_2O, below.

The orientation of the atom is specified by three angles, where alpha is the rotation about the z-axis, beta is the rotation about the transformed x-axis (transformed by alpha), and gamma is the rotation about the transformed z-axis (transformed by both alpha and beta).

The position of the atom is generally only relevant for molecules. However, when rotating the atom using the graphical interface, it is always rotated about the origin. Distances can be specified

in Bohr atomic radii, a_0, where $a_0 = 5.29177249 \times 10^{-11}$ m. See the section on Orbital Mathematics, page 45, for more details.

Molecules are calculated using the Linear Combination of Atomic Orbitals (LCAO) method. This computes the values Ψ for each atom at a point in space, sums them, and then squares the sum to produce Ψ^2. This has the nature that positive phases will reinforce each other, as will negative phases, and that positive phases will cancel out negative phases. The positive phase is where Ψ is positive.

Some examples of different atoms are given in Figure 5. Figure 5a shows a 3s0 that has been cutaway to show the interior structure of the orbital. Figure 5b shows an 11m7 (very energetic), and Figure 5c shows H_2O, which is defined using three atoms. The first is an oxygen atom located at the origin, with n=2, l=1, m=1, 8 protons, mass=16 amu, and a factor of +1. The second is a hydrogen atom located at x=0, y=0.7589 Å, z=0.5877 Å, with n=1, l=0, s=0, 1 proton, mass=1 amu, and a factor of -1. The third is a hydrogen atom located at x=0, y=-0.7589 Å, z=0.5877 Å, with n=1, l=0, s=0, 1 proton, mass=1 amu, and a factor of +1. These individual atoms are shown in Figures 6a, 6b, and 6c.

Figure 5a: 3s0, $\Psi^2=10^{-3.7}$

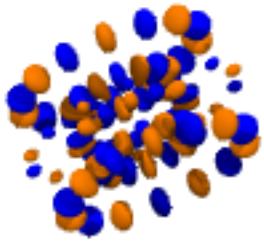


Figure 5b: 11m7, $\Psi^2=10^{-6}$

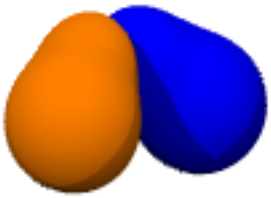


Figure 5c: H_2O, $\Psi^2=10^{-1}$

Figure 6a: 2p1, $\Psi^2=10^{-1}$, 8 protons, mass=16 amu

Figure 6b: 1s0, $\Psi^2=10^{-1}$, factor=-1

Figure 6c: 1s0, $\Psi^2=10^{-1}$, factor=+1

Although the three quantum numbers, *n*, *l*, and *m*, define the shape of the orbital, the actual shape is more dependant on the value *n*-*l*, *l*-|*m*|, and *m*. These quantities determine how many and what kind of asymptotes (where the phase changes) the orbital will have. An atomic orbital always has *n*-*l*-1 spherical asymptotes, (*l*-|*m*|)/2 conical asymptotes (where a 1/2 conical asymptotes is a plane, which can be thought of as a degenerate cone), and |*m*| planar asymptotes. See Figure 7. A more complete example of this is given in the Grand Table (page 55), where all orbitals with $n \leq 10$ are arranged in different manners to illustrate their structure.

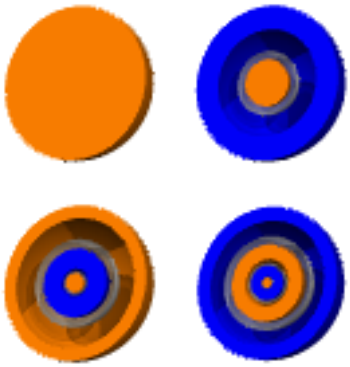


Figure 7a: Cutaway of orbitals showing the spherical asymptotes dependant on *n*-*l*-1. The four orbitals clockwise from the upper left are 1s0, 2s0, 4s0, and 3s0.

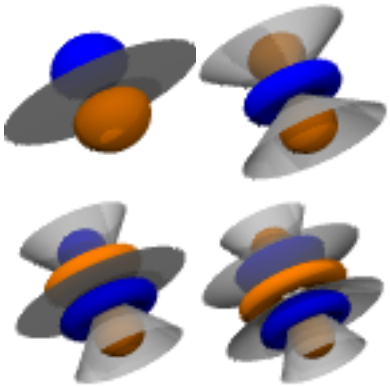


Figure 7b: Orbitals showing the conical asymptotes dependant on *l*-|*m*|. The four orbitals clockwise from the upper left are 2p0, 3d0, 5g0, and 4f0.

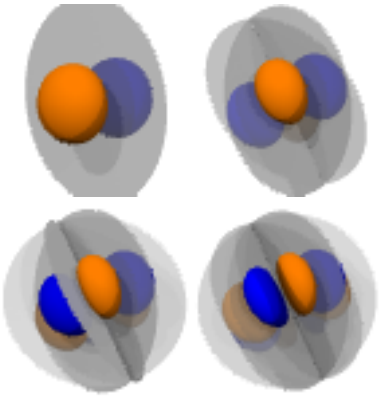


Figure 7c: Orbitals showing the planar asymptotes dependant on *m*. The four orbitals clockwise from the upper left are 2p1, 3d2, 4f3, and 5g4.

LIGHTING

By "illuminating" an orbital, its three-dimensional structure becomes more apparent. This is specified in the **Display / Lighting** menu option. Selecting this displays the dialog shown in Figure 7. Any non-negative number of light sources can be specified.

Figure 8: Lighting Dialog

The location of each light source is specified with respect to the origin of the orbital. This light source can appear to be fixed (always in the same orientation as the camera), or can rotate along with the orbital. The second option is called "rotate with viewpoint".

Light sources do not affect point probability plots. For polygon displays, the light source is always considered a planar source, while for raytracing, it is always a point source. A planar source does not appear to change with position, while a point source does. An example of a point source at different distances from the center of an orbital is shown in Figure 9.


Figure 9a: 4f0, one light source located at -80,80,80 meters


Figure 9b: 4f0, one light source located at -8,8,8 angstroms

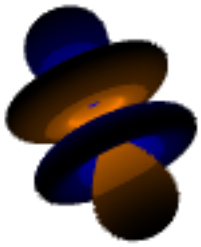


Figure 9c: 4f0, one light source located at -0.8,0.8,0.8 angstroms

The intensity of the light source and the color of the orbital determine the color that is drawn on the screen. If the intensity have a value of 1.00, then a point on the orbital's surface which is normal to the light source will be exactly the specified color. A point on the orbital's surface which is 90° to the light source will be 1/2 of the specified color, and a point on the surface which is 180° different will be black. Light sources are additive, which generally means that the total brightness of all light sources should be around 1. Brightness above 1 can be specified, however, which can be useful with probability density plots.

The ambiance of the light source determines how shadows are computed. A light source with an ambiance of 1.00 will not cast any shadows. An ambiance of zero will produce completely black shadows. See Figure 10.

Figure 10a: One light source at -90,90,90 meters with an intensity of 1 and an ambiance of 1

Figure 10b: One light source at -90,90,90 meters with an intensity of 1 and an ambiance of 0.6

Figure 10c: Two light sources at -90,90,90 meters with an ambiance of 0.6, and at 30,30,70 meters with an ambiance of 0.4, both with an intensity of 0.55

STEREO DISPLAY

In order to facilitate seeing the shape of an orbital, it can be displayed in many different ways. Lighting and color help provide shape information, but an actual three-dimensional display works even better. There are many ways of displaying a three-dimensional display on a standard computer. Most of these require special glasses or hardware, but this is not necessarily an expensive process. Orbital Viewer supports seven different stereo (or three-dimensional) viewing modes, counting the standard monoscopic display. These are selected in the **Display / Stereo** menu option, which displays the dialog shown in Figure 11. Each of the stereo modes is detailed along with information on where to obtain or how to build viewing hardware. Interocular distance and actual separation are discussed under the Stereoscope and Interlaced modes, even though they apply to most of the different modes. Some of the mathematics are discussed in the Imaging Mathematics section, starting on page 52.

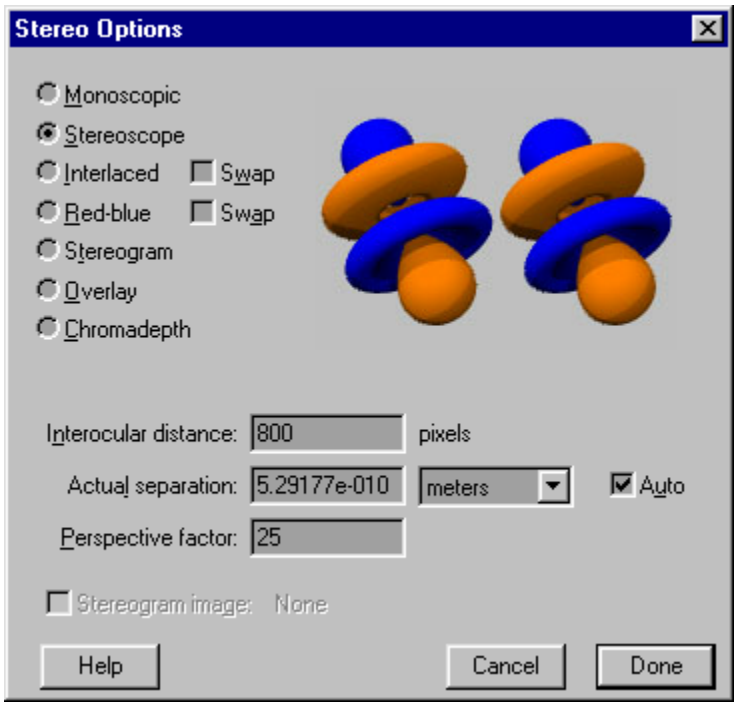


Figure 11: Stereo Options Dialog

All of the figures produced by Orbital Viewer are perspective drawings, with the degree of perspective determined by the perspective value. The perspective value selects the default viewing distance from the orbital. The camera view point is initially located away from the orbital at a distance equal to the perspective value times the diameter of the orbital. The orbital is then sized to fit the screen. This effectively determines the focal length of the "camera" which is used to view the orbital. The smaller the perspective factor, the more distorted the orbital appears. Perspective factors of 25 and larger will all appear very similar. See Figure 12 for examples.

Figure 12a: 4f0 with a perspective factor of 25

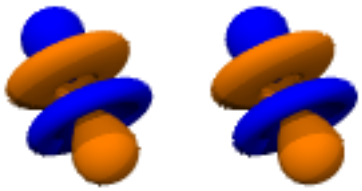
Figure 12b: 4f0 with a perspective factor of 5

Figure 12c: 4f0 with a perspective factor of 1

Monoscopic

The standard display mode is monoscopic. There are no stereo options associated with it.

Stereoscope



One of the easiest ways to see a three-dimensional image is through the use of a stereoscope. Normally, each eye sees a slightly different perspective of a three-dimensional object. To make an object look three dimensional on the computer screen, it is necessary to show each eye a different view. A stereoscope relies on having two different images, one for the left eye and one for the right. This would normally appear on top of one another (see the Overlay mode, below). By using a device called a stereoscope, the images can be displayed side by side. The stereoscope uses mirrors to properly align the images.

Two values affect how the stereoscopic display is drawn. The interocular distance is the separation between the eyes. This is the effective separation based on both the viewer's own geometry and on the dimensions of the stereoscope. It is given in pixels, since it is also dependant on the screen resolution. The actual separation is the distance between the view points of the left and right image. A larger actual separation will increase the depth of the orbital, but will also make it harder to see properly through the stereoscope. The automatic mode uses a separation equal to 1/4 of the orbital's radius.

Building a Stereoscope

A good stereoscope can cost several hundred dollars, mainly due to the expense of the mirrors. However, it is possible to construct a reasonable stereoscope very cheaply using items which are available at many hardware stores. To build a simple stereoscope, the following materials:

- 2 pieces of mirror, 6 x 4 inches
- 2 pieces of mirror, 6 x 3 inches
- 2 pieces of pegboard, 11 x 3 inches, with 1/4 inch holes (holes spaced 1 inch apart)
- 7 dowels, 1/4 inch diameter, 6 1/2 inches long
- sturdy tape (such as strapping tape)
- black tape (such as electrical tape)
- wood glue (optional)

Figure 13 shows the stereoscope. These dimensions can be varied quite a bit, but the spacing between the mirrors should be kept close to those given. For materials, any standard decorative mirror will work. It can often be purchased as 12x12 inch tile and then cut with a glass cutter

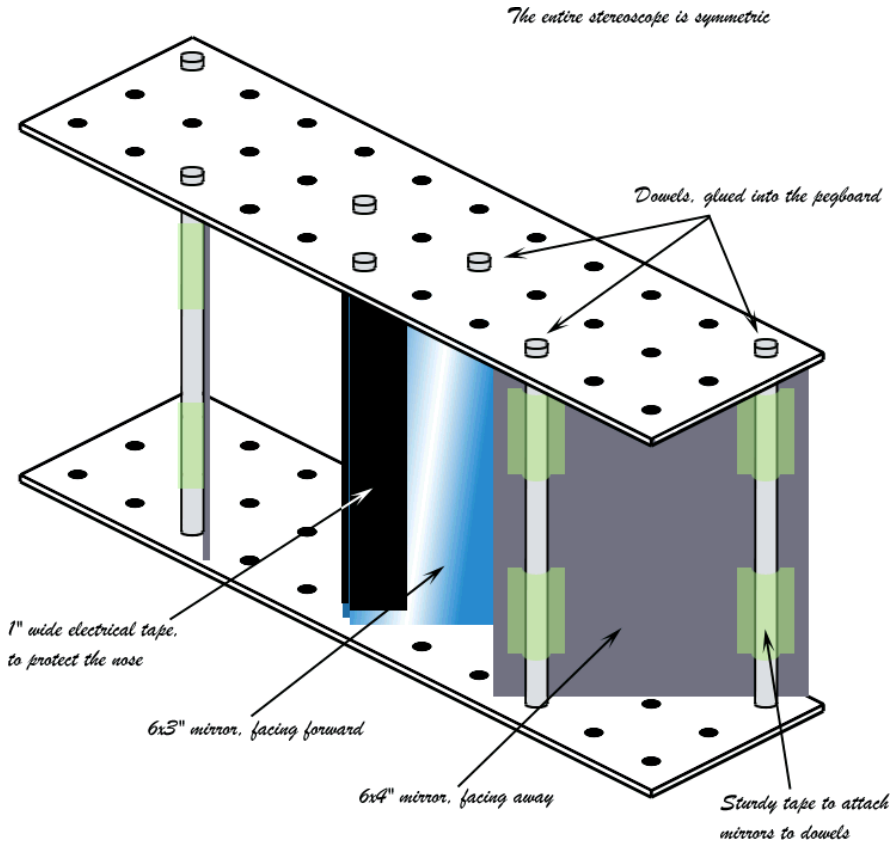


Figure 13: How to build a simple stereoscope

Interlaced

Recently, LCD shutter glasses have become reasonably common. Using these, the computer rapidly alternates between the left and right image. The glasses open and close a shutter so that each eye only sees the image meant for it. On the display, the image is generally stored with the left and right image on alternate lines, and an option exists to swap which lines are used with each side. The hardware takes care of the rest.

As with the stereoscope display, the interlaced display requires an interocular distance. This is actual distance between the viewer's eyes in terms of pixels. Since the size of a pixel is dependant on the monitor and on the graphics mode, this will vary between computers.

Red-Blue

The classic way to view something in three-dimensions is the use of red-blue glasses. This type of image is also referred to as an anaglyph. This works by using a filter to allow the left eye to only see the red image, and the right eye to only see the blue image.

The interocular distance and actual separation work just like the interlaced display mode.

Although glasses can be constructed from red and blue filters, they are generally inexpensive and easy to come by. At the time of this writing, anaglyph glasses could be obtained from several vendors advertising on the world wide web. Prices generally were a few dollars a pair.

Note that the colors don't have to be limited to red and blue. The actual colors used can be modified in the Color dialog.

Stereogram

Recently, the single image random dot stereogram (SIRDS) has been used widely. This is a very simple technique which does not require any special viewing hardware. Unfortunately, many people have difficulty viewing stereograms. The stereogram works by superimposing the images for the left and right eyes on top of each other. Since this would be too hard to see otherwise, each point is colored randomly, and points in the right image and made to match the corresponding point in the left image.

Either a repeating random pattern can be used, or a source image can be selected. This image is repeated horizontally, with the repetition occurring roughly at 1/6 the width of the interocular distance.

The interocular distance is required, and works just like the interlaced display mode. No actual separation is needed since the left and right images are not constructed separately. Instead, a stereogram uses a single depth map, where each the height of each point in the image is used to produce the figure.

Overlay

The overlay technique is essentially the same as the stereoscope method, except that the left and right images are placed so that a stereoscope is not required. Only people who are naturally very good at seeing stereo will be able to merge the two image so that they appear to be three-dimensional.

The interocular distance and actual separation work just like the interlaced display mode.

Chromadepth

Chromadepth is a process which converts color into depth information. This is done through the use of special Chromadepth glasses, which are essentially refraction and diffraction lenses combined into a single plastic film. Parts of the image which are close to the viewer are colored red, while parts which are far away are colored blue, with the colors travelling through the spectrum. The interocular distance and actual separation are not used, since only a single image is employed.

Only the colors from blue to red are used (not purple) due to the nature of computer monitors. To make the spectrum perceptionally uniform, a hexcone color model is used. See the references in *Photogrammetric Engineering and Remote Sensing* for some technical articles the hexcone color space and on the Chromadepth glasses.

Chromadepth is a trademark of Chromatek, Inc. For more information about Chromadepth glasses, contact Chromatek at 11450-F North Fulton Industrial Boulevard, Alpharetta, Georgia 30201-4703, or email them at c3d@chromatek.com.

CAMERA

The orbital is viewed from a specific point in space. This can be considered as a camera which has a position and orientation. The effective focal length is set using the perspective value in the stereo dialog, see page 12. Additionally, the image size can be explicitly set. The camera options are selected in the **Display / Camera** menu option, which displays the dialog shown in Figure 14.

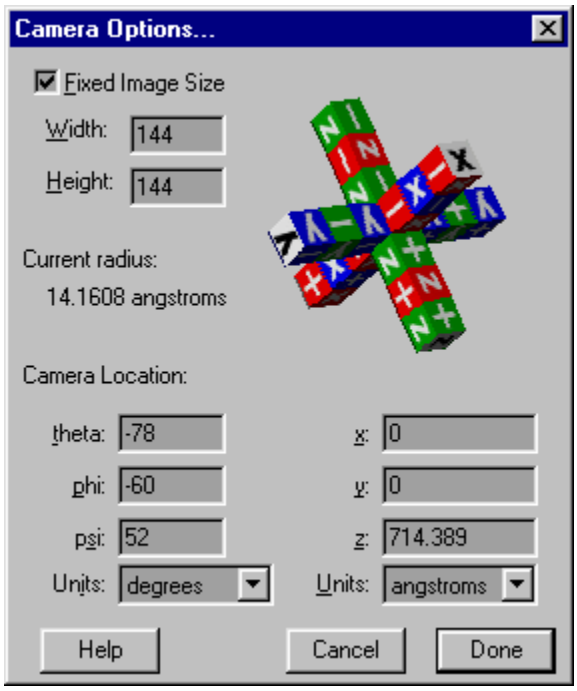


Figure 14: Camera Options Dialog

The image size is normally the same as the current window size. For the command line version of the program, the default size is 640 x 480 pixels. If fixed size is selected, the image will be generated at the requested size, then scaled to fit the window. All copying, saving, and calculations are done at this fixed size. This has the advantage that if the window size changes, a raytraced orbital does not have to be recomputed.

Many of the functions in the program are based on the radius or diameter of the orbital. For a surface probability plot, the radius is the distance from the origin to the outermost point on the selected probability surface. For a probability density plot or for an asymptote plot, the radius is located where a surface of probability would be located if it had a probability of 1/1000th of the maximum probability in the orbital. For molecules, the analyzed radius is sometimes larger than the actual radius would be by this definition.

If the camera is in the reset position, the coordinate system has x heading toward the right of the screen, and y heading vertically up the computer screen. The coordinate system is always right handed. The figure in the Camera Options Dialog shows the orientation, scale, and position of the current camera.

The orientation of the camera is specified by three angles, where theta is the rotation about the z-axis, phi is the rotation about the transformed x-axis (transformed by theta), and psi is the rotation about the transformed y-axis (transformed by both theta and phi). Note that this is slightly different from the angles used to specify atom orientation. In this case, the atom's angles match the angles which are commonly specified in the physics equations used to compute atomic orbitals, while the camera angles match those which are commonly used in photogrammetry for determining image orientation.

The position of the camera is based on the current orientation. It is located in the transformed coordinate system (transformed by theta, phi, and psi). This allows the orientation to be modified while still pointing the camera at the center of the orbital. The camera will point directly at the center of the orbital if it has an x and y coordinate of zero.

CUTAWAY

It is often difficult to see the internal structure of an orbital. To facilitate viewing, a cutaway can be defined which removes part of the orbital. The cutaway options are selected in the **Display / Cutaway** menu option, which displays the dialog shown in Figure 15.

The cutaway can either be a plane, a corner, or a wedge, which removes 1/2, 1/4, or 1/8 of the orbital, respectively. This can also be inverted, leaving only 1/2, 1/4, or 1/8 of the orbital. Inverting the cutaway leaves exactly what would have been taken away without the inversion. See Figure 16.

Figure 15: Cutaway Options Dialog

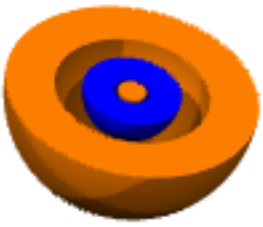


Figure 16a: 3s0, planar cutaway

Figure 16b: 3s0, corner cutaway

Figure 16c: 3s0, wedge cutaway

Figure 16d: 3s0, inverted wedge cutaway

When a probability surface is plotted, the cutaway can either create a surface at the cutaway, or it can leave the surface "open" which allows a sort of view into the orbital. See Figure 17.

The orientation and position of the cutaway are specified in the orbital's coordinate system. The orientation of the cutaway is specified by three angles, where alpha is the rotation about the z-axis, beta is the rotation about the transformed x-axis (transformed by alpha), and gamma is the rotation about the transformed z-axis (transformed by both alpha and beta). The position of the cutaway determines exactly which part of the orbital will be removed. It can be located anywhere in space.

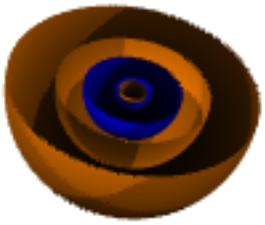


Figure 17: 3s0, planar cutaway (as in Figure 16a) without cutaway surface

RENDERING METHOD

Orbital Viewer can draw orbitals in any of three ways: point density plots, polygon surface of probability approximations, and raytracing. In the graphical version of Orbital Viewer, it is often useful to have a lower quality image which can be readily manipulated, which is then replaced by a higher quality image once the orbital has been selected. The render options are selected in the **Display / Render** menu option, which displays the dialog shown in Figure 18.

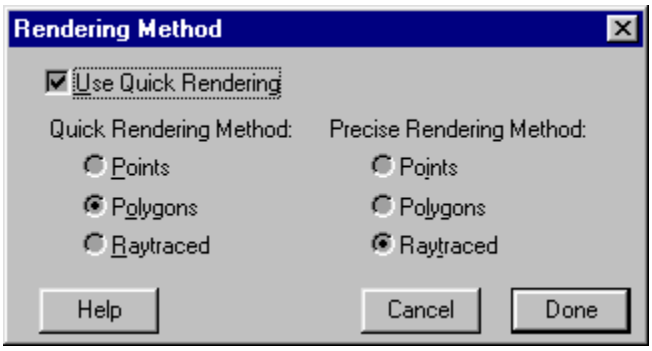


Figure 18: Rendering Method Dialog

If quick rendering is turned off, only the precise method is used. If it is turned on, then the orbital will be drawn in the quick mode first followed by the precise mode.

POINT DRAWING MODE

Point drawing mode is generally the fastest for the computer to draw. It is not necessarily the fastest to compute. The number of points used in a point plot can be set by selecting the **Display / Point Options** menu, see Figure 19. A point drawing is always a probability density plot, with more points located in parts of the orbital with a higher probability. The points are computed by choosing a point at random, and keeping it only if a randomly generated number is lower than the probability at that point.

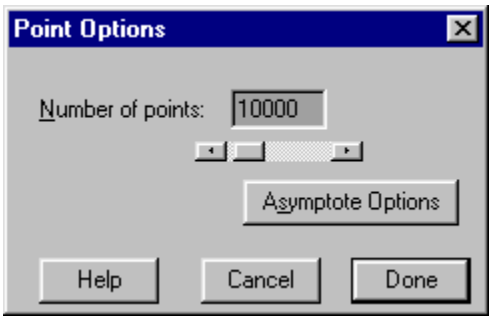


Figure 19: Point Options Dialog

An example of a point drawing is shown in Figure 20. The size of the points can be modified in the Preferences Dialog.

Any positive number of points can be generated, provided that the computer has enough memory. Lighting does not effect the color of the point, only phase. Note that if the resulting data is saved as a VRML file, some VRML readers can only handle a few thousand points.

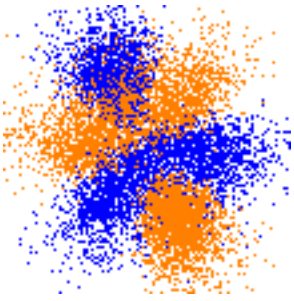


Figure 20: 4f0 point probability plot with 10,000 points

POLYGON DRAWING MODE

The polygon drawing mode is a surface of probability display. It is generally fast to compute and fast to manipulate. Options for this mode are specified by selecting the **Display / Polygon Options** menu. Selecting this displays the dialog shown in Figure 21.

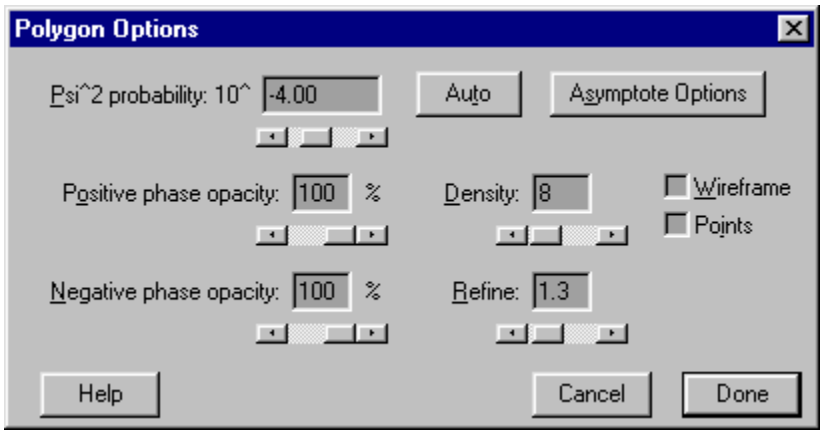


Figure 21: Polygon Options Dialog

Since this displays a surface of constant probability, this probability is specified using Ψ^2. This is specified as a logarithmic number, since it is often very small. A probability which is more negative will produce a surface further from the center of the orbital. If the probability is too positive, all or some of the orbital may not appear. See Figure 22.

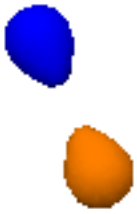


Figure 22a: 4f0, $\Psi^2=10^{-3.6}$

Figure 22b: 4f0, $\Psi^2=10^{-3.8}$

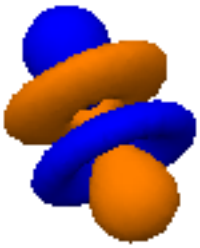


Figure 22c: 4f0, $\Psi^2=10^{-4.0}$

Figure 22d: 4f0, $\Psi^2=10^{-4.7}$

The Auto button will select a value which produces a pleasant figure. This option also adjusts the density (see below) to the minimum value needed to probably compute the surface.

An opacity of 100 % produces a solid surface. An opacity of 0 % will completely hide the surface. Opacities in between these two values allow rear portions of the orbital to be seen. See Figure 23. The opacity is simulated on the screen by only drawing the specified percentage of the pixels which would normally appear. Both the front and back surface of the orbital have this opacity, so that a opacity of 50 % will effectively block 75 % (all but (50 %)^2) of the view of what is behind it. To use true opacity, raytracing must be used, see page 24.

Figure 23: 4f0, $\Psi^2=10^{-4.0}$, both phases at 50 % opacity

To compute the polygons used in the surface of the orbital, a two step process is used. First, the space from the center of the orbital out to the maximum radius of the orbital is divided into a set of tetrahedrons. If the

probability along any leg of a tetrahedron crosses through the specified value of Ψ^2, then a vertex is located at the exact crossing point. If two vertices are too close together, they are combined. This prevents triangles of very small size from being created. This number of tetrahedrons is affected by the density parameter. After a set of triangles defining the surface is created, the triangles are refined. This is done by dividing long edges of the triangle into two, creating a vertex and generally changing two triangles into four. The new vertex is "snapped" to the selected surface of probability. This process continues until no triangle leg is too long based on the refinement parameter.

For computation reasons, the density must be an even integer that is at least six. In order to accurately compute the surface of probability, the density should be the minimum of $(n\text{-}l)*2$, $(l\text{-}|m|)*2$, and $|m|*2$. If the surface has holes or is too rough, increasing the density will fix it. Generally densities which are more than twice the minimum value, as specified above, do not produce substantially different results. The density will also determine how well a cutaway appears in a polygon surface plot. For examples of different density settings, see Figure 24.

Figure 24a: 5g1 with a density of 6 and a refinement of 0

Figure 24b: 5g1 with a density of 8 and a refinement of 0

Figure 24c: 5g1 with a density of 12 and a refinement of 0

Refining the polygons has two major effects. It makes the all the polygons roughly the same size, and it smoothes the surface of the orbital. A refinement of zero will not change the polygons as generated by the selected density. A non-zero refinement ensures that all triangles have sides shorter than the orbital's radius divided by the density and divided by the refinement parameter. Examples of different refinements are given in Figure 25. Doubling the refinement quadruples the number of triangles.

Figure 25a: 5g1 with a density of 8 and a refinement of 0

Figure 25b: 5g1 with a density of 8 and a refinement of 1.0

Figure 25c: 5g1 with a density of 8 and a refinement of 2.0

There are three styles in which a set of polygons can be drawn. Normally, the polygons are solid, with the color based on the light source. They can also be drawn as a wireframe, or as just the vertices (points). The wireframe mode best shows what actual polygons are used to make up a surface. The point mode produces something like the point density plot, but with a surface instead. See Figure 26.

Figure 26a: Solid 5g1

Figure 26b: Wireframe 5g1

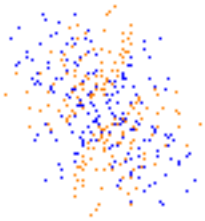


Figure 26c: Point 5g1

RAYTRACING DRAWING MODE

The raytrace drawing mode has the most options and is the highest quality. It is also the slowest. Options for this mode are specified by selecting the **Display / Raytrace Options** menu. Selecting this displays the dialog shown in Figure 27. This mode can display both surfaces of probability, like the polygon drawing mode, and probability density plots, like the point drawing mode.

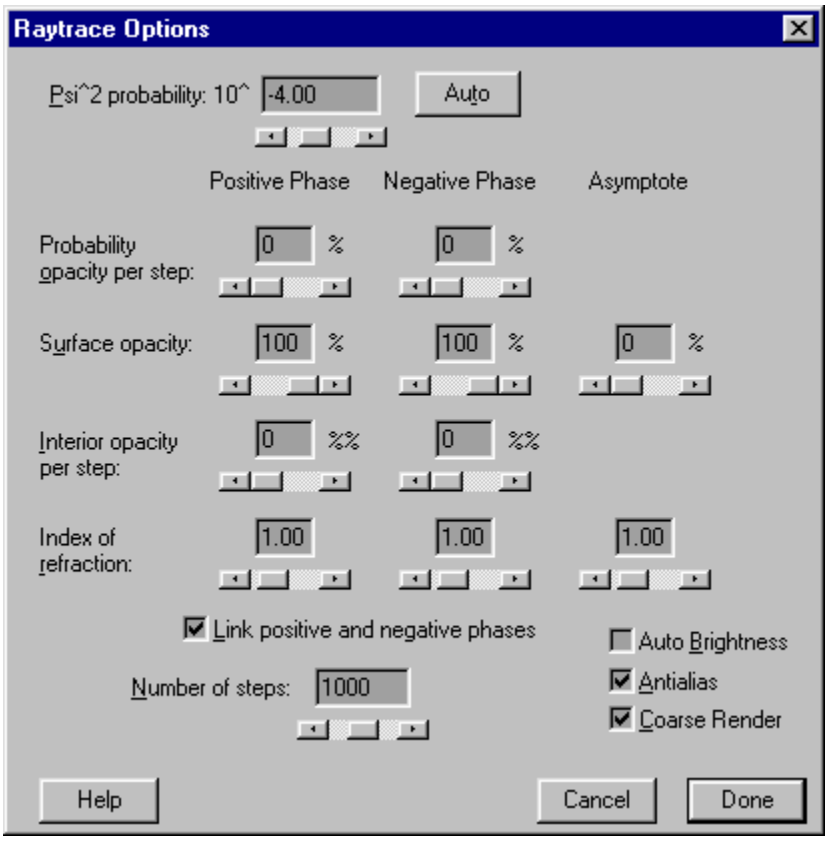


Figure 27: Raytrace Options Dialog

If a surface of constant probability is displayed (determined by the surface and interior opacity), this probability is specified using Ψ^2. This is specified in the manner detailed under the section on Polygon Drawing Mode, see page 21 and also see Figure 22.

To create a probability density plot, the probability opacity must be non-zero. To compute the color of any pixel, a ray is tracked from the camera along the appropriate line in space. This is done in a series of finite steps, with the specified number of steps across the orbital's radius. The probability is evaluated at each step. If the probability of a step has the maximum probability in the orbital, then the opacity of that step is the specified probability opacity. If the probability is zero, then that step is entirely transparent. Similarly, all steps' opacity is directly proportional to the probability at that step. This opacity calculation technique is also to compute how much light is provided by each light source. As such, shadows consume a large amount of computing power when used with probability opacity. If no light sources are used, the plot resembles actual measurements of atomic orbitals, such as those made through laser techniques. See Figure 28 for some examples.

Figure 28a: 4f0, 5 % probability density plot with no light sources

Figure 28b: 4f0, 5 % probability density plot with one light source

Figure 28c: 4f0, 10 % probability density plot with one light source

If the surface probability is non-zero, the surface specified by the Ψ^2 value will be drawn. The opacity calculations treat this surface as being infinitely thin (since it is). As such, the opacity does not change based on the viewing angle. The opacity affects both the front and back surface of the orbital, such that a 50 % opacity, will only transmit 25 % of the what is behind that lobe of the orbital. See Figure 29 for examples.

Figure 29a: 4f0, 25 % surface opacity

Figure 29b: 4f0, 50 % surface opacity

Figure 29c: 4f0, 100 % surface opacity

In addition to drawing the positive and negative phases of the orbital, the asymptotes can also be shown. This is similar to drawing a surface of probability with $\Psi^2=0$. Although the asymptotes are infinitely thin, the opacity is drawn such that it is proportional to the viewing angle, such that an asymptote which is oblique in the picture will appear thicker. See Figure 30.

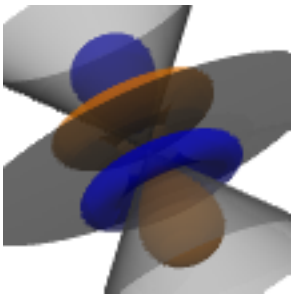


Figure 30: 4f0, 50 % surface opacity, 30 % asymptote surface opacity

The interior opacity settings are used to provide similar effects in the positive and negative phases, where the thickness of the phase dictates the opacity of the orbital. The interior opacity is set as a %% value, where a value of 1 %% = 0.0001. As with the probability opacity, the orbital is analyzed in a series of finite steps, with the specified number of steps across the orbital's radius. If a point is within the surface of probability based on the Ψ^2 value, then that step has the specified opacity. As with probability opacity, using light sources with

shadows vastly increases the amount of computation required. Some examples are shown in Figure 31.

Figure 31a: 4f0, 10 %% interior opacity with one light source

Figure 31b: 4f0, 25 %% interior opacity with one light source

Figure 31c: 4f0, 50 %% interior opacity with one light source

Another effect which can be applied to the orbital is an index of refraction. Ordinary space has an index of refraction of 1.00. The orbital can appear to be made of "glass" by giving it a higher index, or, the orbital can be a cavity within a "glass" space by using an index less than 1. When an index of refraction is applied to a probability density plot, the specified index exists at the point of highest probability. This index approaches 1 as the probability approaches zero. The asymptotes are treated as having a thickness equal to the orbital's radius divided by the specified step size for purposes of refracting light. Some examples are shown in Figure 32.

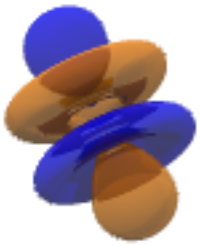


Figure 32a: 4f0, 0.9 index of refraction, 50 % surface opacity. Compare to Figure 29b

Figure 32b: 4f0, 1.1 index of refraction, 50 % surface opacity

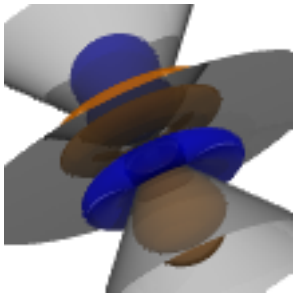


Figure 32c: 4f0, 1.1 index of refraction, 50 % surface opacity, 30 % asymptote opacity. Compare to Figure 30

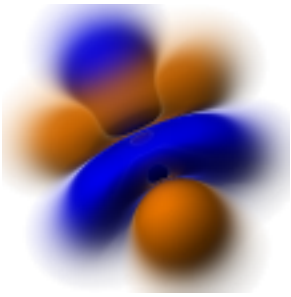


Figure 32d: 4f0, 2.0 index of refraction, 5 % probability opacity. Compare to Figure 28b

In all of the figures above, the positive and negative phases have been treated equally. This is not required. By setting one phase's probability to zero, only the other phase will be shown. This can also be used to show one phase with a probability density plot and the other with a surface of probability.

Probability density plots, asymptotes plots, and interior probability plots all work by stepping along a ray for each pixel. The probability is analyzed at each point, and the opacity, direction of the ray, and color is evaluated. The number of steps specified is the number taken along the radius of the orbital. This number can be varied, which results in more or less accurate displays. It has a

direct effect on the time it takes to compute the orbital. If light sources with shadows are used, doubling the number of steps will square the amount of computations required. To keep the same total opacity, as the number of steps decreases, it is necessary to increase the specified opacity. Examples with different step sizes are shown in Figure 33. In all cases $(1\text{-}opacity)^{steps}$ is a constant.

Figure 33a: 4f0, 92.3 % probability density plot with 20 steps

Figure 33b: 4f0, 40.1 % probability density plot with 100 steps

Figure 33c: 4f0, 5.0 % probability density plot with 1000 steps

The orbital is first drawn very coarsely and then refined. This allows an initial view of the orbital much more quickly than would otherwise be obtained from raytracing. The first pass is drawn with points spaced every 16x16 pixels. Auto brightness can be applied at this point. The second pass draws all points spaced every 4x4 pixels. The points in the first pass are not recomputed. A third pass draws every pixel, but some may be approximated if coarse rendering is turned on. If antialiasing is used, a fourth pass analyzes the orbital down to 1/4 pixel resolution.

The color of each pixel is dependant on the selected orbital and the selected light source. If a low opacity is used, the resultant picture can be very dim. Specifying auto brightness will adjust the color such that the brightest values will be close to the maximum possible. The necessary brightness is calculated by coarsely drawing the orbital at normal brightness, then adjusting it so that the coarse drawing will be at a good brightness. This may permit pixels which are oversaturated (too bright) if there are features which are too small to show up on the coarse drawing.

Coarse rendering can be much faster than complete rendering, since not all the points must be independently computed. After each point on a 4x4 pixel grid is computed, the remaining points are drawn. If the four points on the 4x4 grid are similar enough, then the pixels within that 4x4e region are interpolated from the corners. Similarity is defined as being within 15 color units of each other in all three (red, green, and blue) color channels. There are 256 color units available to each pixel (each can range from 0 to 255).

Antialiasing can be applied after every pixel is computed. If adjacent pixels are non similar, then that region of the orbital is computed using an oversampling technique. This is done to an accuracy of 1/4 pixel. This means that near a color transition, up to 16 times as many calculations are required to compute an antialised pixel. Two points are considered similar if they are within 4 color units of each other in all three color channels. Figure 34 shows an orbital with and without antialiasing.

Figure 34a: 4f0, without antialiasing

Figure 34b: 4f0, with antialiasing

ASYMPTOTE DRAWING MODE

With any of the other drawing modes, the orbital's asymptotes can also be drawn. This is done simultaneously in the raytracing drawing mode. For point and polygon drawing modes, the asymptotes are computed as a set of polygons. Options for this are specified by selecting the **Display / Asymptote Options** menu. Selecting this displays the dialog shown in Figure 35. This options are essentially identical to those for the polygon drawing mode. See page 21 for more information.

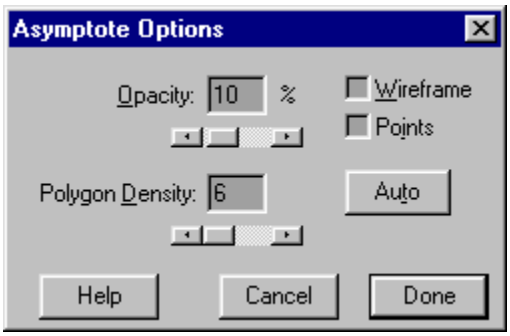


Figure 35: Asymptote Options Dialog

SEQUENCES

One of the interesting ways to view an orbital or to observe how a given parameter affects an orbital is to draw a sequence of frames and create an animation of it. This can be done using the **Display / Play Sequence** menu item, which displays the dialog shown in Figure 36.

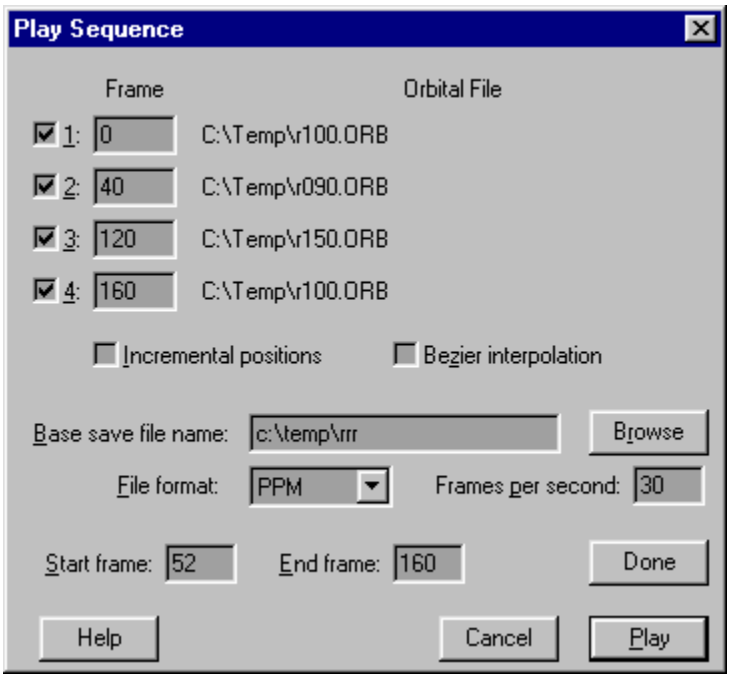
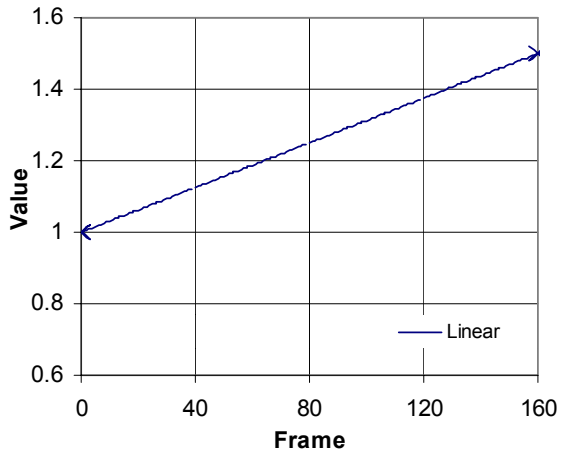


Figure 36: Play Sequence Dialog

Two, three, or four orbitals can be used to specify the sequence. These orbitals can have any desired values for any parameters. Each of the specification orbitals also has an integer frame number associated with it. The value of the orbital along with the frame number is used to determine the value which will be used during the sequence. Although these are requested as files in the graphical interface version of the program, the actual sequence specification is stored in a single file along with the standard orbital information. See the section on input file formats, page 31, for more information.

Interpolation

The sequence can start and end on any integer frame values. When creating the sequence, all frames between the starting and ending frames, inclusive, are drawn. For each frame, the orbital is determined by interpolating from the two, three, or four orbital specification files.



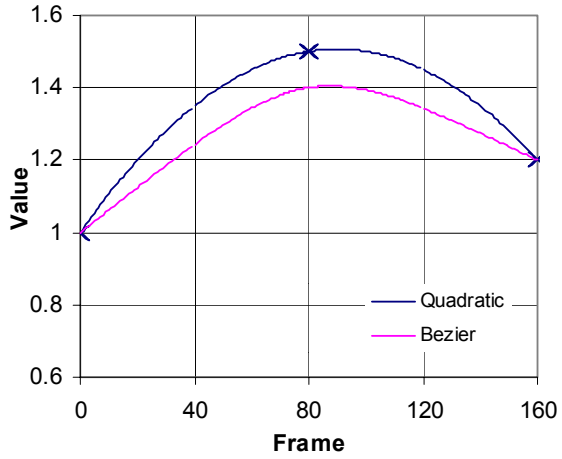
Graph 1: Linear interpolation between a value of 1 at frame 0 and 1.5 at frame 160

If only two orbitals are used in the sequence specification, the interpolation is linear.

Linear interpolation satisfies the equation

$$v = Af + B \tag{1}$$

where v is the interpolated value, f is the frame number, and A and B are constants determined based on satisfying the equation for the values and frame numbers from the orbital specification files. An example of linear interpolation is shown in Graph 1. The Bezier setting has no effect on linear interpolation.



Graph 2: Quadratic and three-point Bezier interpolations between a value of 1 at frame 0, 1.5 at frame 80, and 1.2 at frame 160

If three orbitals are used in the sequence specification, the interpolation can be either quadratic or a three-point Bezier spline.

Quadratic interpolation satisfies the equation

$$v = Af^2 + Bf + C \tag{2}$$

where v is the interpolated value, f is the frame number, and A, B, and C are constants determined based on satisfying the equation for the values and frame numbers from the orbital specification files. This is always a parabola.

A three-point Bezier spline satisfies the simultaneous equations

$$\begin{aligned} f &= Ap^3 + Bp^2 + Cp + D \\ v &= A'p^3 + B'p^2 + C'p + D' \end{aligned} \tag{3}$$

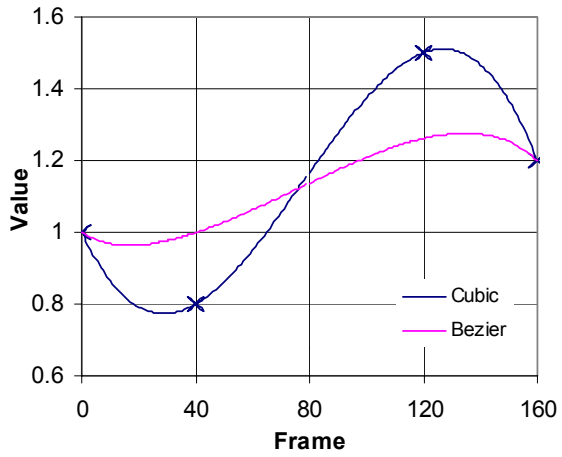
where p is a parametric value ranging from 0 to 1, v is the interpolated value, and f is the frame number. The values A, B, C, D, A', B', C', and D' are defined as

$$\begin{aligned} A &= f_3 - f_1 & A' &= v_3 - v_1 \\ B &= -3f_2 + 3f_1 & B' &= -3v_2 + 3v_1 \\ C &= 3f_2 - 3f_1 & C' &= 3v_2 - 3v_1 \\ D &= f_1 & D' &= v_1 \end{aligned} \tag{4}$$

where f_1, f_2, and f_3 are the frame numbers and v_1, v_2, and v_3 are the values of the three specification points.

The curve produced by the Bezier spline always lies entirely within the triangle formed by the three specification points. The slope of the Bezier spline at the first point is the same as that of a line between the first and middle points, and the slope at the last point is the same as a line between the last and middle points.

An example and comparison of quadratic interpolation and three-point Bezier interpolation are shown in Graph 2.



Graph 3: Cubic and four-point Bezier interpolations between a value of 1 at frame 0, 0.8 at frame 40, 1.5 at frame 120, and 1.2 at frame 160

If four orbitals are used in the sequence specification, the interpolation can be either cubic or a four-point Bezier spline.

Cubic interpolation satisfies the equation

$$v = Af^3 + Bf^2 + Cf + D \tag{5}$$

where v is the interpolated value, f is the frame number, and A, B, C, and D are constants determined based on satisfying the equation for the values and frame numbers from the orbital specification files.

A four-point Bezier spline satisfies the simultaneous equations specified in Equation (3), differing in that the values A, B, C, D, A', B', C', and D' are defined as

$$\begin{aligned} A &= f_4 - 3f_3 + 3f_2 - f_1 & A' &= v_4 - 3v_3 + 3v_2 - v_1 \\ B &= 3f_3 - 6f_2 + 3f_1 & B' &= 3v_3 - 6v_2 + 3v_1 \\ C &= 3f_2 - 3f_1 & C' &= 3v_2 - 3v_1 \\ D &= f_1 & D' &= v_1 \end{aligned} \tag{6}$$

where f_1, f_2, f_3, and f_4 are the frame numbers and v_1, v_2, v_3, and v_4 are the values of the four specification points. This is identical to the three-point version if the middle two specification points are equal.

The curve produced by the Bezier spline always lies entirely within the quadrilateral formed by the four specification points. The slope of the Bezier spline at the first point is the same as that of a line between the first and second points, and the slope at the last point is the same as a line between the third and last points.

An example and comparison of cubic and four-point Bezier interpolations are shown in Graph 3.

When a frame is calculated which lies outside of the specification range, the values are extrapolated using the appropriate linear, quadratic, cubic, or Bezier function. The Bezier function is not uniquely defined outside of the specification range, and is unpredictable in this extrapolation. The order of the specification orbitals has no effect on linear, quadratic, or cubic interpolation. It does, however, have a significant effect on Bezier interpolation.

Incremental Position

If incremental position is turned on, then the camera location and orientation is not interpolated using the above functions. All other values are interpolated, including cutaway and lighting positions and orientations. Instead of interpolating the camera position, the camera position is determined by using the only the first and second specification orbitals. The position and orientation is calculated using the equation

$$v = v_1 + (t - t_1)(v_2 - v_1) \tag{7}$$

where t is the current frame, t_1 is the frame number of the first specification orbital, v_1 is the position or orientation value of the first specification orbital, v_2 is the position or orientation value of the second specification orbital, and v is the position or orientation value for the current frame. Note that the frame number of the second specification orbital is not used in this calculation

The incremental position mode provides an easier way to rotate an orbital. Interpolation between angles does not necessarily result in the desired output since all angles are in the range of $[0,2\pi)$ radians or [0,360) degrees. The difference between the second and first specification orbitals is used as a step for each frame.

Interpolated Values

The number of atoms and light sources will be the minimum of any orbital specification file.

The following parameters are interpolated between frames:

Colors – background, positive phase, negative phase, and asymptote
Camera – position, orientation, and fixed image width and height
Atoms – mass, n, l, m, number of protons, factor, position, and orientation
Light sources – intensity, ambiance, position, and orientation
Asymptote – opacity and density
Cutaway – position and orientation
Points – number
Polygons – Ψ^2, density, refinement, and opacity
Raytrace – Ψ^2, probability opacity, surface opacity, interior opacity, index of refraction, number of steps
Stereo display – interocular distance, separation, and perspective

The following parameters are copied from the first orbital specification:

Light sources – local
Cutaway – type, surface, and inversion
Raytrace – auto brightness, antialias, and coarse rendering
Stereo – mode, auto separation, and stereogram image

Sequence Output

A sequence generates a series of frames. These pictures can be saved. The graphical version of Orbital Viewer can save each frame as a binary PPM, TIF, or BMP file, or can save the entire sequence as an AVI file. The command line version of Orbital Viewer outputs ASCII PPM or VRML files. See the section of output file formats, page 42, for more details.

If a sequence is saved as a binary image file (PPM, TIF, or BMP), the file name is determined by appending the frame number to the base file name and the file type as an extension. For example, if the base file name is C:\TEMP\ORB, and the sequence is being saved as PPM files, then frame 26 will be stored in the file C:\TEMP\ORB26.PPM. For AVI files, the file name is only determined using the base file name and the file type extension. In the preceding example, this would produce the file C:\TEMP\ORB.AVI.

When an AVI file is created, it is created in an uncompressed format. The number of frames per second can be specified. If the file already exists, frames will be appended to it. If the file does not have frames of the same width, height, and frames per second, the sequence is not saved. Uncompressed AVI files can be compressed on some systems using the **Display / Compress AVI** menu function.

See Figure 37 for some sample frames from a sequence.

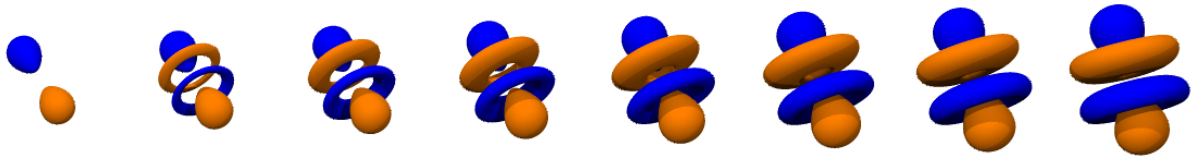


Figure 37: Frames 68 through 75 from a sequence which varies camera orientation and Ψ^2 of a 4f0 orbital

FILE FORMATS - INPUT

Orbital Viewer can read two different file formats. Orbital specification files use the .ORB extension, and contain all the information necessary to create an orbital or orbital sequence. Orbital Viewer files use the .OV extension, and contain the same information as the .ORB files, plus additional binary information which allows computation to be resumed from the point at which it was saved.

The .OV files are version dependant, and will be treated exactly like .ORB files if the version of Orbital Viewer which created them is not the same as the version used to read them. They are always treated like .ORB files with the command line version of Orbital Viewer. The binary part of the .OV files is not discussed in this manual. For information on this file format, contact the author.

Orbital Specification Files (.ORB)

Orbital Specification files are plain text (ASCII) files which contain a header phrase, followed by a list of keywords with each keyword followed either by a number, by a single phrase, or by braces {} contain another list of keywords. For example, the keyword "`CameraCenterZ(m)`" might be followed by the number "`7.14388974643e-008`", the keyword "`RenderMode`" by the phrase "`Raytrace`", and the keyword "`Atom`" by "`{ n 4 l f m 0 }`", where "`n`", "`l`", and "`m`" are themselves keywords.

Keywords, numbers, phrases, and braces are all separated by white space (tabs, spaces, carriage returns, line feeds, etc.). This white space is arbitrary.

Unless otherwise noted, keywords can be in any order within the file or within a set of braces {}. Also unless otherwise noted, the keyword should not be repeated. There are no drawbacks to repeating a keyword except that only the last instance of the keyword will actually be used. All keywords are optional.

Case of keywords and phrases does not matter.

Numbers can be specified as integers (e.g., "`6`"), rationals (e.g., "`6.5`"), exponentials (e.g., "`6.5e-3`"), or as hexadecimal (e.g., "`0x3C`"). Hexadecimal numbers are always prefixed by "`0x`". Any form can be used at any time a number is required after a keyword.

In some instances, phrases can be free form text. In this case, the phrase is surrounded by double quotes "", and all characters are allowed within the quotes except for more quotes. Any phrase can be surrounded by quotes, if desired.

Comments can be included after any white space by starting them with a number sign (#). Everything after the number sign is ignored until a carriage return or line feed is encountered. Comments can not occur within phrases surrounded by double quotes.

The file begins with the header phrase "`OrbitalFileV1.0`". Orbital Viewer (.OV) files differ in that they begin with the header phrase "`OrbitalViewerFileV1.0`".

Below is a list of all keywords and the arguments which must follow them:

`AsymptoteColor` - (number) - the 0xRRGGBB color of to use for asymptotes. This is in the range of 0 to 16777215 (0xFFFFFF), or -1 to use the default color. Example: 0x808080.

`Asymptotes` - (keyword list) - this has the following keywords and arguments:

`Density` - (number) - the density of asymptote polygons. This is an even integer greater than or equal to 6.

`Opacity` - (number) - the opacity of the asymptote, between 0 and 1.

`Style` - (phrase) - `Solid` | `Wireframe` | `Points`

`Atom` - (keyword list) - the atom keyword can be repeated any number of times. For a molecular orbital, there will be at least two `Atom` statements. This has the following keywords and arguments:

`AngleAlpha(rad)` - (number) - atom orientation angle alpha in radians.

`AngleBeta(rad)` - (number) - atom orientation angle beta in radians.

`AngleGamma(rad)` - (number) - atom orientation angle gamma in radians.

`CenterX(m)` - (number) - position of atom's center along the x axis in meters.

`CenterY(m)` - (number) - position of atom's center along the y axis in meters.

`CenterZ(m)` - (number) - position of atom's center along the z axis in meters.

`Factor` - (number) - probability multiplicand factor, usually -1 or 1.

`l` - (number) - orbital quantum number, *l*. Integer between 0 and *n*-1.

`m` - (number) - angular momentum quantum number, *m*. Integer between -*l* and +*l*.

`Mass(kg)` - (number) - mass of atom in kilograms.

`n` - (number) - principal quantum number, *n*. Positive integer. On most computers this must be less than or equal to 30.

`Neutrons(N)` - (number) - number of neutrons in atom. This is a non-negative integer.

`Protons(Z)` - (number) - number of neutrons in atom. This is a positive integer.

`BackgroundColor` - (number) - the 0xRRGGBB color of to use for the background. This is in the range of 0 to 16777215 (0xFFFFFF), or -1 to use the default color. Example: 0x000000.

`CameraCenterX(m)` - (number) - position of camera's center along the transformed x axis in meters. The transformation is by the camera angles theta, phi, and psi.

`CameraCenterY(m)` - (number) - position of camera's center along the transformed y axis in meters. The transformation is by the camera angles theta, phi, and psi.

`CameraCenterZ(m)` - (number) - position of camera's center along the transformed z axis in meters. The transformation is by the camera angles theta, phi, and psi.

`CameraCx` - (number) - this is the camera's focal length divided by the pixel size. It is a unitless quantity, and is typically in the 1000-10000 range.

`CameraPhi(rad)` - (number) - camera orientation angle phi in radians.

`CameraPsi(rad)` - (number) - camera orientation angle psi in radians.

`CameraTheta(rad)` - (number) - camera orientation angle theta in radians.

`Cutaway` - (keyword list) - this has the following keywords and arguments:

`AngleAlpha(rad)` - (number) - cutaway orientation angle alpha in radians.

`AngleBeta(rad)` - (number) - cutaway orientation angle beta in radians.

`AngleGamma(rad)` - (number) - cutaway orientation angle gamma in radians.

`Invert` - (phrase) - `No` | `Yes`

`PositionX(m)` - (number) - cutaway center position along the x axis in meters.

`PositionY(m)` - (number) - cutaway center position along the y axis in meters.

`PositionZ(m)` - (number) - cutaway center position along the z axis in meters.

`Surface` - (phrase) - `No` | `Yes`

`Type` - (phrase) - `None` | `Plane` | `Corner` | `Wedge`

`DefaultPerspective` - (number) - perspective factor to switch to when the reset position function is selected. This is a positive number.

`EndOfFile` - (any) - all data after this keyword and argument is ignored. The argument can have any value, but it must exist. This is used in Orbital Viewer .OV files to signal the end of the orbital specification and the beginning of the binary segment.

`FileName` - (phrase) - this is typically the full pathname of the file, but it can be any text. Example: `"C:\ORB\PDF\FIG10C.ORB"`

`FixedHeight` - (number) - the number of pixels in height of a fixed sized camera image. If `FixedSize` is set to `Yes`, this should be at least 2.

`FixedSize` - (phrase) - `No` | `Yes`

`FixedWidth` - (number) - the number of pixels in width of a fixed sized camera image. If `FixedSize` is set to `Yes`, this should be at least 2.

`Frame` - (number) - this is the starting/current frame of a sequence. If this is within the keyword `Sequence`'s braces, then it is the frame number for that sequence specification. It can be any integer.

`FramesPerSecond` - (number) - this is the number of frames per second stored in an AVI file created by a sequence. It is a positive integer.

`LastFrame` - (number) - the ending frame of a sequence. This can be any integer.

`LastHeight` - (number) - the height in pixels of the last image drawn in a window. This is the height of the window if a fixed camera size is not used or raytracing is not the current rendering mode. Otherwise, this is the fixed height.

`LastWidth` - (number) - the width in pixels of the last image drawn in a window. This is the width of the window if a fixed camera size is not used or raytracing is not the current rendering mode. Otherwise, this is the fixed width.

`Light` - (keyword list) - the light keyword can be repeated any number of times. This has the following keywords and arguments:

`Ambience` - (number) - the ambiance of this light source, between 0 and 1.

`Intensity` - (number) - the intensity of the light source. This is a nonnegative number, typically between 0 and 1.

`Local` - (phrase) - `No` | `Yes`

`PositionX(m)` - (number) - light source position along the x axis in meters.

`PositionY(m)` - (number) - light source position along the y axis in meters.

`PositionZ(m)` - (number) - light source position along the z axis in meters.

`NegativeColor` - (number) - the 0xRRGGBB color of to use for negative phases. This is in the range of 0 to 16777215 (0xFFFFFF), or -1 to use the default color. Example: 0xFF8000.

`Perspective` - (number) - this is the current perspective factor. The orbital will fill the screen when the camera is at a distance equal to this number multiplied by the orbital's diameter. This is a positive number.

`Points` - (keyword list) - this has the following keywords and arguments:

`Points` - (number) - this is the number of points generated in a point density plot. It is a positive integer.

`Polygons` - (keyword list) - this has the following keywords and arguments:

`Density` - (number) - the initial density of the polygons. This is an even integer greater than or equal to 6.

`NegativeOpacity` - (number) - the opacity of the negative phase, between 0 and 1.

`PositiveOpacity` - (number) - the opacity of the positive phase, between 0 and 1.

`Refinement` - (number) - polygons will be refined until the length of all triangle legs is shorter than the orbital's radius divided by the density and divided by this parameter. This is a non-negative number.

`Style` - (phrase) - `Solid` | `Wireframe` | `Points`

`PositiveColor` - (number) - the 0xRRGGBB color of to use for positive phases. This is in the range of 0 to 16777215 (0xFFFFFF), or -1 to use the default color. Example: 0x0000FF.

`PreviewColor` - (number) - the 0xRRGGBB color of to use for the lighting and cutaway previews. This is in the range of 0 to 16777215 (0xFFFFFF), or -1 to use the default color. Example: 0x8080FF.

`Psi^2(log10)` - (number) - the factor used to determine which surface of probability is drawn in polygon mode and when the surface opacity is non-zero in raytracing mode. This is typically a negative number, but can be any value.

`QuickRenderMode` - (phrase) - `Points` | `Polygons` | `Raytrace`

`Raytrace` - (keyword list) - this has the following keywords and arguments:

`Antialias` - (phrase) - `No` | `Yes`

`AsymptoteOpacity` - (number) - the opacity of the asymptote, between 0 and 1.

`AsymptoteRefraction` - (number) - the index of refraction of the asymptote. This is a positive number.

`AutoBrightness` - (phrase) - `No` | `Yes`

`Coarse` - (phrase) - `No` | `Yes`

`NegativeRefraction` - (number) - the index of refraction of the negative phase. This is a positive number.

`NegInteriorOpacity` - (number) - the interior opacity of the negative phase per step, between 0 and 1.

`NegProbOpacity` - (number) - the probability opacity of the negative phase per step, between 0 and 1.

`NegSurfaceOpacity` - (number) - the surface opacity of the negative phase, between 0 and 1.

`PosInteriorOpacity` - (number) - the interior opacity of the positive phase per step, between 0 and 1.

`PositiveRefraction` - (number) - the index of refraction of the positive phase. This is a positive number.

`PosProbOpacity` - (number) - the probability opacity of the positive phase per step, between 0 and 1.

`PosSurfaceOpacity` - (number) - the surface opacity of the positive phase, between 0 and 1.

`Steps` - (number) - the number of steps used in computing probability and interior opacity based figures. This is a positive number.

RenderMode - (phrase) - Points | Polygons | Raytrace

Scale(m) - (number) - the scale of the orbital. This is the diameter of the orbital in meters. It is used to determine how far to pan or zoom and orbital when the mouse is used to manipulate it.

Sequence - (keyword list) - the sequence keyword can be repeated from two to four times. If there is only one Sequence keyword, an actual sequence is not defined. Sequences use exactly the same keywords as the entire orbital file, except they do not recognize the Sequence keyword (no recursion).

SequenceBase - (phrase) - this is the full path and root name of the sequence output file(s). Example: "C:\TEMP\SEQ"

SequenceBezier - (phrase) - No | Yes

SequenceFileType - (phrase) - PPM | TIF | BMP | AVI

SequenceIncrement - (phrase) - No | Yes

Stereo - (keyword list) - this has the following keywords and arguments:

AutoSeparation - (phrase) - No | Yes

BlueColor - (number) - the 0xRRGGBB color of to use for the 'blue' color of RedBlue mode stereo images. This is in the range of 0 to 16777215 (0xFFFFFF), or -1 to use the default color. Example: 0x0000FF.

ImageName - (phrase) - this is the full pathname of an image file used for stereograms. Example: "C:\ORB\DATA\PATTERN.JPG". It is not used with the command line version of Orbital Viewer.

Interocular - (number) - the interocular distance in pixels for all stereo modes except the stereoscope mode. This is a non-negative number.

Mode - (phrase) - Monoscopic | Stereoscope | Interlaced | RedBlue | Stereogram | Overlay | Chromadepth

RedColor - (number) - the 0xRRGGBB color of to use for the 'red' color of RedBlue mode stereo images. This is in the range of 0 to 16777215 (0xFFFFFF), or -1 to use the default color. Example: 0x0000FF.

RedSide - (phrase) - Left | Right

Separation(m) - (number) - The separation between the left and right view points in meters. This is a positive number.

StereoscopeInter - (number) - the interocular distance in pixels for the stereoscope mode. This is a non-negative number.

TopScanline - (phrase) - Left | Right

`UseImage` - (phrase) - `No` | `Yes` This is not used with the command line version of Orbital Viewer.

`UseQuickRender` - (phrase) - `No` | `Yes`

Example File

This is a complete orbital file as saved by Orbital Viewer. It is the orbital specification for Figure 10c.

```
OrbitalFileV1.0
DefaultPerspective                      25
BackgroundColor       0xFFFFFF
UseQuickRender        Yes
QuickRenderMode       Raytrace
RenderMode            Raytrace
FixedSize             Yes
FixedWidth            144
FixedHeight           144
FileName              "C:\ORB\pdf\fig10a.orb"
Scale(m)                2.85755707985e-009
Perspective                             25
LastWidth                              144
LastHeight                             144
CameraCenterX(m)                         0
CameraCenterY(m)                         0
CameraCenterZ(m)        7.14388974643e-008
CameraTheta(rad)            -1.36135709286
CameraPhi(rad)              -1.04719769955
CameraPsi(rad)              0.907571196556
CameraCx                     3239.99975586
Atom {      # Atom 1
  n                     4
  l                     f
  m                     0
  Neutrons(N)           0
  Protons(Z)            1
  Mass(kg)                    1.6605402e-027
  }
Light {     # Light 1
  PositionX(m)                          -90
  PositionY(m)                           90
  PositionZ(m)                           90
  Intensity                             0.55
  Ambience                               0.6
  Local                 No
  }
Light {     # Light 2
  PositionX(m)                           30
  PositionY(m)                           30
  PositionZ(m)                           70
  Intensity                             0.55
  Ambience                               0.4
  Local                 No
  }
Psi^2(log10)                            -4
Cutaway {     # Cutaway 1
  Type                  None
  Surface               Yes
  Invert                No
  }
Stereo {
  Mode                  Monoscopic
  TopScanline           Left
  RedSide               Left
  AutoSeparation        Yes
  UseImage              No
  Interocular                           250
  StereoscopeInter                      800
  Separation(m)              5.29177249e-010
  }
Points {
  Points                10000
  }
Polygons {
  Density               8
  Refinement                            1.3
  PositiveOpacity                         1
  NegativeOpacity                         1
  Style                 Solid
  }
Asymptotes {
  Density               0
  Opacity                                 0
  Style                 Solid
  }
Raytrace {
  PosProbOpacity                          0
  PosSurfaceOpacity                       1
  PosInteriorOpacity                      0
  NegProbOpacity                          0
  NegSurfaceOpacity                       1
  NegInteriorOpacity                      0
  AsymptoteOpacity                        0
  PositiveRefraction                      1
  NegativeRefraction                      1
  AsymptoteRefraction                     1
  Steps                 1000
  AutoBrightness        No
  Coarse                No
  Antialias             Yes
  }
```

FILE FORMATS - OUTPUT

Orbital Viewer can output files in several different formats. The graphical version of Orbital Viewer can output orbital specification files (.ORB), Orbital Viewer files (.OV), binary PPM files, TIFF files, BMP files, AVI files, VRML version 1 files, or partial Digistar II files. The command line version of Orbital Viewer outputs either ASCII PPM files or VRML version 1 files.

Orbital Specification Files (.ORB)

This file format is fully detailed in the section on input file formats. See page 35.

Orbital Viewer Files (.OV)

This file format is version specific, and is treated exactly like an orbital specification file (.ORB) if it was created by a different version of Orbital Viewer. The start of the file is identical to a .ORB file except for the header phrase. After an ASCII description of the orbital, a binary record follows. This binary record stores the current state of the calculations done on the orbital by Orbital Viewer. For example, it stores the polygons computed in polygon drawing mode. This eliminates the need to recalculate these values at the expense of file storage space. For more information on this binary format, please contact the author.

Portable Pixel Map Files (.PPM)

The portable pixel map file format was invented by Jef Pozkanzer. This is a widely used format on Unix machines, and is pretty much as simple as a format can get. There are six possible modes for a PPM file, but Orbital Viewer only uses two of them. These are color ASCII and color binary formats. If the command line version of Orbital Viewer is used to generate a raytraced orbital, the output will always be in ASCII PPM format.

For an ASCII color PPM file, the file begins with the two letters "P3" this is followed by the width and height of the image, then the maximum value a pixel can have (typically 255), then the red, green, and blue values of every pixel. All values are in ASCII decimal notation and are separated by white space. For example, the beginning of a PPM file might look like "P3 144 128 255 0 4 234 2 8 233 ...". Pixels are stored from the top left and are always in the order RGB. The file can be slightly more complicated, but Orbital Viewer adheres to this simplest version.

A binary PPM also begins with two letters, "P6" followed by the width, height, and maximum pixel value in ASCII. Exactly one character separates the maximum pixel value from a binary record of the image. This binary record has three bytes per pixel in the order RGB.

More information can typically be found in Unix man pages and by searching for Jef Pozkanzer's web site (which seems to move around a bit).

TIFF Files (.TIF)

The Tagged Image File Format is an extremely versatile, flexible format with many, many options. Orbital Viewer stores TIF images as 24-bit RGB images using run-length encoding. The complete

specification can be obtained from the Adobe Corporation, and, at the time of this writing was located at http://www.adobe.com/supportservice/devrelations/PDFS/TN/TIFF6.pdf . Orbital Viewer is entirely baseline TIFF compliant.

Bitmap Files (.BMP)

Bitmap files are an inefficient file format which is native to Windows. Most books on Windows cover the format. Orbital Viewer stores images in type 40 bitmaps with 24-bits per pixel. These files consist of a 14 byte file header followed by a 40 byte image header. The image header contains (amongst other things), the image width and height and the color mode. The actual pixels are stored bottom row first and in BGR (not RGB) format. Each line is padded to a multiple of four bytes (not an even multiple of pixels). This format is also used somewhat with AVI files.

VRML Files (.WRL)

Virtual Reality Markup Language files are used for storing three-dimensional information. This is only used for polygon and point drawing modes. Orbital Viewer is entirely compliant with VRML version 1. In the graphics interface version of the program, three options can be set. These determine how color is specified within the VRML file since different readers behave differently. Some VRML readers can not handle more than a few thousand points. The VRML specification can be obtained at http://www.vrml.org/Specifications . If the command line version of Orbital Viewer is used to generate a point density or polygon orbital, the output will always be in VRML format.

AVI Files (.AVI)

AVI files are used to store video sequences. The actual file format is a subset of RIFF (Resource Interchange File Format), which was originally created for the Amiga computer. Although Orbital Viewer outputs a valid AVI file, it is not the only way such a file could be constructed. The AVI file format is poorly documented, but information can be found in the book *Multimedia and ODBC API Bible* by Richard J. Simon, ISBN 1-57169-011-5.

A brief description of the AVI files created by Orbital Viewer is given below.

Byte	Item
0x0000	"RIFF"
0x0004	length of file excluding first 8 bytes
0x0008	"AVI "
0x000C	"LIST"
0x0010	length of list record (0xC0)
0x0014	"hdrl"
0x0018	"avih"
0x001C	length of avi header (0x38)
0x0020	microseconds per frame
0x0024	bytes per second
0x0028	granularity (0)
0x002C	flags (0x810)
0x0030	number of frames
0x0034	0
0x0038	1
0x003C	bytes per frame

0x0040	width in pixels
0x0044	height in pixels
0x0048	0
0x0058	"LIST"
0x005C	length of list record (0x74)
0x0060	"strl"
0x0064	"strh"
0x0068	length of stream header (0x38)
0x006C	"vids"
0x0070	"DIB "
0x0074	0
0x0080	scale (1)
0x0084	frames per second
0x0088	starting frame (0)
0x008C	number of frames
0x0090	bytes per frame
0x0094	quality (2000 for some reason)

0x0098	0
0x00A0	width in pixels (2 bytes)
0x00A2	height in pixels (2 bytes)
0x00A4	"strf"
0x00A8	length of frame header (0x28)
0x00AC	standard bitmap header (0x28 bytes)
0x00D4	"JUNK"
0x00D8	length of padding (0xF18)
0x00DC	0
0x0FF4	"LIST"
0x0FF8	length of list record (4+(8+bytes per frame)*number of frames)
0x0FFC	"movi"

0x1000	"00db"
0x1004	bytes per frame
0x1008	actual frame data. This is the bitmap without the header
...	Previous 3 entries repeated for all frames
0x1000+(8+bytes per frame)*number of frames)	
+0x0000	"idx1"
+0x0004	length of index (0x10*number of frames)
+0x0008	"00db"
+0x000C	0x10
+0x0010	(4+(8+bytes per frame)*frame number)
+0x0014	bytes per frame
...	Previous 4 entries repeated for all frames

Digistar II Files (.TXT)

The Digistar II is a projector used in planetariums. This output option only works with point and polygon drawing modes. If any points have been calculated, the points are output. Otherwise, the vertices of the polygons are output. A set of points are stored in the file, each have a set of coordinates and a brightness. The brightness is always set to 1.

ORBITAL MATHEMATICS

This section is a copy of an older paper on Schrödinger's Equation. This is the equation used for all orbital computations. Molecular orbitals are computed using the linear combination of atomic orbitals (LCAO) technique, where the value of Ψ for each atom is added before the result is squared to produce Ψ^2.

Introduction

Schrödinger's Equation is converted from the traditional form $H\Psi = E\Psi$ in spherical coordinates to a general form which does not contain any derivatives, only summations. It is assumed that some knowledge of Schrödinger's Equation already exists. All symbols used are identified. The result is presented in a spherically-separable format, and the spherical gradient for the function is shown. These results are valid for all hydrogenic atoms, regardless of quantum number, nuclear mass, or atomic number.

List of Symbols

H quantum mechanical Hamiltonian

Ψ wave function

E internal energy

$\hbar$ Planck's constant = $m_e a_0^2 / \tau = h/2\pi = 1.0545726 \times 10^{-34}\ \mathrm{kg\,m^2\,s^{-1}}$

a_0 radius of first Bohr orbit = $\hbar^2 / m_e e^2 = 5.29177249 \times 10^{-11}\ \mathrm{m}$

a radius of first orbit = $\hbar^2 / \mu e^2$

μ reduced mass of the electron = $m_e M / (m_e + M)$

m_e mass of electron = $9.109389427 \times 10^{-31}\ \mathrm{kg}$

m_Z mass of proton = $1.6726230 \times 10^{-27}\ \mathrm{kg}$

m_N mass of neutron = $1.6749286 \times 10^{-27}\ \mathrm{kg}$

M mass of the nucleus

τ time for electron to travel 1 radian in first Bohr orbit = $2.41888440 \times 10^{-17}\ \mathrm{s}$

e negative of charge on electron = $m_e^{1/2} a_0^{3/2} / \tau = 1.518907291 \times 10^{-14}\ \mathrm{kg^{1/2}\,m^{3/2}\,s^{-1}}$

$\hat{e}$ natural logarithm base = 2.71828182846

i unit imaginary number

∇ del operator

V potential energy

n principal quantum number (1, 2, 3, ...)

l orbital quantum number (0, 1, ..., $n-2$, $n-1$)

m angular momentum quantum number, usually m_l, ($-l$, $-l+1$, ..., $l-1$, l)

s spin quantum number, usually m_s, (±1/2)

r radial distance

θ rotation from z axis (0 to π)

ϕ rotation around z axis (0 to 2π)

Z atomic number

Ψ_{nlm} electron probability function

R_{nl} radial probability function

Y_{lm} angular probability function

Θ_{lm} probability function from z axis

Φ_m probability function around z axis

ρ radial factor

L_n^m associated Laguerre polynomial

L_n Laguerre polynomial

P_l^m associated Legendre polynomial

Electron Orbital Wave Function

The solution of Schrödinger's equation

$$H\Psi = E\Psi, \tag{1}$$

where $H = -\frac{\hbar^2}{2\mu}\nabla^2 + V(r)$, for the hydrogenic atom, in which $V(r) = -\frac{Ze^2}{r}$ is the potential energy due to charge, is

$$\Psi_{nlm}(r,\theta,\phi) = \frac{R_{nl}(r)}{r} Y_{lm}(\theta,\phi). \tag{2}$$

The radial part of this function is

$$\frac{R_{nl}(r)}{r} = \frac{1}{r}\sqrt{\frac{Z(n-l-1)!}{n^2 a\,(n+l)!^3}}\;\hat{e}^{-\frac{\rho}{2}}\rho^{l+1}\;L_{n+l}^{2l+1}(\rho), \tag{3}$$

where $\rho = \frac{2Zr}{na}$, $a = \frac{\hbar^2}{\mu e^2}$, and $L_{n+l}^{2l+1}(\rho)$ is the associated Laguerre polynomial. μ is the reduced mass of the system, which is $\mu = m_e M/(m_e + M)$, where m_e is the mass of an electron and M is the nuclear mass. The associated Laguerre polynomial is

$$L_n^m(x) = \left(\frac{d}{dx}\right)^m L_n(x), \tag{4}$$

where the Laguerre polynomial is

$$L_n(x) = (-1)^n\left[x^n - \frac{n^2}{1!}x^{n-1} + \frac{n^2(n-1)^2}{2!}x^{n-2} - \frac{n^2(n-1)^2(n-2)^2}{3!}x^{n-3} + \cdots + (-1)^n n!\right]. \tag{5}$$

The associated Laguerre polynomial can be rewritten

$$L_n^m(x) = \left(\frac{d}{dx}\right)^m \sum_{j=0}^{n}\left((-1)^{n-j}\frac{n!^2}{(n-j)!^2\;j!}x^{n-j}\right) = \sum_{j=0}^{n}\left((-1)^{n-j}\frac{n!^2}{(n-j)!^2\;j!}\left(\frac{d}{dx}\right)^m x^{n-j}\right) \tag{6}$$

However, $\left(\frac{d}{dx}\right)^m x^n = \frac{n!}{(n-m)!}x^{n-m}$ when $m \le n$, and $\left(\frac{d}{dx}\right)^m x^n = 0$ when $m > n$, assuming n and m are both non-negative integers.. Thus

$$L_n^m(x) = \sum_{j=0}^{n-m}\left(\frac{(-1)^{n-j}\,n!^2\;x^{n-m-j}}{(n-j)!\,j!(n-m-j)!}\right) = n!^2\sum_{j=0}^{n-m}\left(\frac{(-1)^{n-j}\,x^{n-m-j}}{(n-j)!\,j!(n-m-j)!}\right). \tag{7}$$

Substituting equation (7) into equation (3) yields

$$\frac{R_{nl}(r)}{r} = \frac{\hat{e}^{-\frac{\rho}{2}}\rho^{l+1}(n+l)!}{rn}\sqrt{\frac{Z(n-l-1)!}{a\,(n+l)!}}\;\sum_{j=0}^{n-l-1}\left[\frac{(-1)^{n+l-j}\,\rho^{n-l-j-1}}{(n+l-j)!\,j!(n-l-j-1)!}\right]. \tag{8}$$

Note that $(-1)^{n+l-j} = (-1)(-1)^{n-l-j-1}$. If the value for ρ is substituted into equation (8), the total radial probability is seen to be

$$\frac{R_{nl}(r)}{r} = -\frac{\hat{e}^{-\frac{Zr}{na}}2^{l+1}Z^{l+1}r^l}{n^{l+2}a^{l+1}}\sqrt{\frac{Z(n-l-1)!(n+l)!}{a}}\;\sum_{j=0}^{n-l-1}\left[\frac{\left(-\frac{2Zr}{na}\right)^{n-l-j-1}}{(n+l-j)!\,j!(n-l-j-1)!}\right]. \tag{9}$$

The angular probability function can be rewritten $Y_{lm}(\theta,\phi) = \Theta_{lm}(\theta)\Phi_m(\phi)$. The probability function from the *z* axis is

$$\Theta_{lm}(\theta) = (-1)^{\frac{m+|m|}{2}} \sqrt{\frac{(2l+1)(l-|m|)!}{2(l+|m|)!}}\, P_l^{|m|}(\cos\theta) \tag{10}$$

where $P_l^{|m|}(\cos\theta)$ is the associated Legendre polynomial. $(-1)^{\frac{m+|m|}{2}}$ is an arbitrary phase, as suggested by Condon and Shortley and used throughout much of the literature. The associated Legendre polynomial is

$$P_l^m(x) = \frac{\left(1-x^2\right)^{\frac{m}{2}}}{2^l l!}\left(\frac{d}{dx}\right)^{l+m}\left[\left(x^2-1\right)^l\right] = \frac{\left(1-x^2\right)^{\frac{m}{2}}}{2^l l!}\left(\frac{d}{dx}\right)^{l+m}\sum_{j=0}^{l}\left(\frac{l!(-1)^j}{j!(l-j)!}x^{2(l-j)}\right). \tag{11}$$

As was done with the associated Laguerre polynomial, the Legendre polynomial is also equal to

$$P_l^m(x) = \frac{\left(1-x^2\right)^{\frac{m}{2}}}{2^l}\sum_{j=0}^{\frac{l-m}{2}}\left(\frac{(-1)^j(2l-2j)!}{j!(l-j)!(l-|m|-2j)!}x^{l-m-2j}\right) \tag{12}$$

where if $l-m$ is an odd number, the upper bound of the summation is $(l-m-1)/2$. This results in a total probability function from the *z* axis of

$$\Theta_{lm}(\theta) = (-1)^{\frac{m+|m|}{2}} \sqrt{\frac{(2l+1)(l-|m|)!}{2(l+|m|)!}}\,\frac{\sin^{|m|}\theta}{2^l}\sum_{j=0}^{\frac{l-|m|}{2}}\left[\frac{(-1)^j(2l-2j)!}{j!(l-j)!(l-|m|-2j)!}\cos^{l-|m|-2j}\theta\right]. \tag{13}$$

The probability function around the *z* axis is

$$\Phi_m(\phi) = \frac{1}{\sqrt{2\pi}}\hat{e}^{im\phi} \tag{14}$$

where $\hat{e}$ is the natural logarithm base and i is the unit imaginary number. A real valued function can be constructed such that

$$\begin{aligned}
\Phi_m(\phi) &= \frac{\sin(m\phi)}{\sqrt{\pi}} && m>0 \\
\Phi_0(\phi) &= \frac{1}{\sqrt{2\pi}} && \text{for} \quad m=0 \\
\Phi_m(\phi) &= \frac{\cos(m\phi)}{\sqrt{\pi}} && m<0
\end{aligned} \tag{15}$$

Note that the spin quantum number, s, does not affect the physical shape of the probability function. Relativistic effects are not included in the preceding equations. These effects are comparatively small. For non-hydrogenic atoms, additional terms must be introduced because of

the interaction between electrons. Only approximations of the many-electron atom can be obtained. Once such approximation is the standard perturbation method, which can be found in much of the literature.

The total wave function for a hydrogenic atom is given below. The terms which are constant with respect to coordinates are grouped together.

$$\begin{aligned}\Psi_{nlm}(r,\theta,\phi) = (-1)^{\frac{m+|m|+2}{2}} & \frac{Z^{l+1}}{n^{l+2}a^{l+1}}\sqrt{\frac{Z(n-l-1)!(n+l)!(l-|m|)!(2l+1)}{\pi a(l+|m|)!}} \\ & \bullet\ r^l \hat{e}^{-\frac{Zr}{na}} \sum_{j=0}^{n-l-1}\left[\frac{(-2Zr/na)^{n-l-j-1}}{(n+l-j)!\,j!(n-l-j-1)!}\right] \\ & \bullet\ \sin^{|m|}\theta \sum_{j=0}^{\frac{l-|m|}{2}}\left[\frac{(-1)^j(2l-2j)!}{j!(l-j)!(l-|m|-2j)!}\cos^{l-|m|-2j}\theta\right] \\ & \bullet\ \begin{cases}\sqrt{2}\sin(m\phi) & (m>0)\\ 1 & (m=0)\\ \sqrt{2}\cos(m\phi) & (m<0)\end{cases}\end{aligned} \tag{16}$$

Orbital Function Gradient

The gradient function in spherical coordinates is

$$\nabla\Psi = \frac{\partial\Psi}{\partial r}e_r + \frac{1}{r}\frac{\partial\Psi}{\partial\theta}e_\theta + \frac{1}{r\sin\theta}\frac{\partial\Psi}{\partial\phi}e_\phi \tag{17}$$

where e_r, e_θ, and e_ϕ are the unit vectors in the spherical coordinate system. This can be rewritten as

$$\nabla\Psi = \nabla_{e_r}\Psi e_r + \nabla_{e_\theta}\Psi e_\theta + \nabla_{e_\phi}\Psi e_\phi \tag{18}$$

The factor in the e_r direction is

$$\nabla_{e_r}\Psi = (-1)^{\frac{m+|m|+2}{2}} \frac{Z^{l+1}}{n^{l+2}a^{l+1}} \sqrt{\frac{Z(n-l-1)!(n+l)!(l-|m|)!(2l+1)}{\pi a(l+|m|)!}}$$

$$\bullet \left[\begin{array}{l} \left(\frac{l}{r} - \frac{Z}{na}\right) r^l \hat{e}^{-\frac{Zr}{na}} \sum_{j=0}^{n-l-1} \left[\frac{\left(-2Zr/na\right)^{n-l-j-1}}{(n+l-j)!\, j!(n-l-j-1)!} \right] \\ -\frac{2Z}{na} r^l \hat{e}^{-\frac{Zr}{na}} \sum_{j=0}^{n-l-2} \left[\frac{\left(-2Zr/na\right)^{n-l-j-2}}{(n+l-j)!\, j!(n-l-j-2)!} \right] \end{array} \right]$$

$$\bullet \sin^{|m|}\theta \sum_{j=0}^{\frac{l-|m|}{2}} \left[\frac{(-1)^j (2l-2j)!}{j!(l-j)!(l-|m|-2j)!} \cos^{l-|m|-2j}\theta \right] \qquad (19)$$

$$\bullet \begin{cases} \sqrt{2}\sin(m\phi) & (m>0) \\ 1 & (m=0) \\ \sqrt{2}\cos(m\phi) & (m<0) \end{cases}$$

where a summation with an upper bound less than zero is evaluated as zero. The factor in the e_θ direction is

$$\nabla_{e_\theta}\Psi = (-1)^{\frac{m+|m|+2}{2}} \frac{Z^{l+1}}{n^{l+2}a^{l+1}} \sqrt{\frac{Z(n-l-1)!(n+l)!(l-|m|)!(2l+1)}{\pi a(l+|m|)!}}$$

$$\bullet\, r^l \hat{e}^{-\frac{Zr}{na}} \sum_{j=0}^{n-l-1} \left[\frac{\left(-2Zr/na\right)^{n-l-j-1}}{(n+l-j)!\, j!(n-l-j-1)!} \right]$$

$$\bullet \frac{1}{r} \left[\begin{array}{l} |m|\cos\theta \sin^{|m|-1}\theta \sum_{j=0}^{\frac{l-|m|}{2}} \left[\frac{(-1)^j (2l-2j)!}{j!(l-j)!(l-|m|-2j)!} \cos^{l-|m|-2j}\theta \right] \\ -\sin^{|m|+1}\theta \sum_{j=0}^{\frac{l-|m|-1}{2}} \left[\frac{(-1)^j (2l-2j)!(l-|m|-2j)}{j!(l-j)!(l-|m|-2j)!} \cos^{l-|m|-1-2j}\theta \right] \end{array} \right] \qquad (20)$$

$$\bullet \begin{cases} \sqrt{2}\sin(m\phi) & (m>0) \\ 1 & (m=0) \\ \sqrt{2}\cos(m\phi) & (m<0) \end{cases}$$

The last component, e_ϕ, is

$$\nabla_{e_\phi}\Psi = (-1)^{\frac{m+|m|+2}{2}} \frac{Z^{l+1}}{n^{l+2}a^{l+1}} \sqrt{\frac{Z(n-l-1)!(n+l)!(l-|m|)!(2l+1)}{\pi a(l+|m|)!}}$$

$$\bullet\ r^l \hat{e}^{-\frac{Zr}{na}} \sum_{j=0}^{n-l-1} \left[\frac{\left(-2Zr/na\right)^{n-l-j-1}}{(n+l-j)!\, j!(n-l-j-1)!} \right]$$

$$\bullet\ \sin^{|m|}\theta \sum_{j=0}^{\frac{l-|m|}{2}} \left[\frac{(-1)^j (2l-2j)!}{j!(l-j)!(l-|m|-2j)!} \cos^{l-|m|-2j}\theta \right] \tag{21}$$

$$\bullet\ \frac{1}{r\sin\theta} \begin{cases} \sqrt{2}\, m\cos(m\phi) & (m>0) \\ 0 & (m=0) \\ -\sqrt{2}\, m\sin(m\phi) & (m<0) \end{cases}$$

IMAGING MATHEMATICS

In addition to Schrödinger's equation, a variety of other equations are used in the calculation of an orbital.

Camera Equations

The viewpoint is treated as a camera with 10 parameters. These parameters are the location of the camera (X_0, Y_0, and Z_0), the orientation of the camera (θ, ϕ, and ψ), the focal length of the camera as compared to the pixel size in horizontal and vertical directions (C_x and C_y), and the center of the camera within the image (x_0 and y_0).

The location of the camera is specified in the rotated coordinate system. The coordinate system is rotated using the equation

$$\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} = \begin{bmatrix} X_0' \\ Y_0' \\ Z_0' \end{bmatrix}^t \begin{bmatrix} \sin\phi'\sin\psi\sin\theta' + \cos\psi\cos\theta' & \sin\phi'\sin\psi\cos\theta' - \cos\psi\sin\theta' & \cos\phi'\sin\psi \\ \cos\phi'\sin\theta' & \cos\phi'\cos\theta' & -\sin\phi' \\ \sin\phi'\cos\psi\sin\theta' - \sin\psi\cos\theta' & \sin\phi'\cos\psi\cos\theta' + \sin\psi\sin\theta' & \cos\phi'\cos\psi \end{bmatrix} \quad (1)$$

where the primed coordinates, X_0', Y_0', and Z_0', are the coordinates specified in the Orbital Viewer program. Similarly, the orientations θ', ϕ', and ψ are as specified in the program, with $\theta' = \theta + \pi$ and $\phi' = \phi + \pi$. The angles are taken differently than in the camera equations given in equation (3), below, for convenience in the program.

Theoretically, there is a focal length, f, and a physical pixel size (λ_x by λ_y). However, the pixel size is always coupled with the focal length, and therefore $C_x = f/\lambda_x$ and $C_y = f/\lambda_y$. The camera always has square pixels ($C_x = C_y$) and is always centered on the image ($x_0 = w/2$, $y_0 = h/2$, where w and h are the width and height of the image).

If the location of three-dimensional point is known, then the location of the corresponding pixel within the camera image is also known. Let X, Y, and Z be the location of the three-dimensional point, and let x and y be the location of the pixel within the image. The relationship is given by the equations

$$\begin{aligned} x - x_0 &= \frac{-C_x\left((X-X_0)m_{11} + (Y-Y_0)m_{12} + (Z-Z_0)m_{13}\right)}{(X-X_0)m_{31} + (Y-Y_0)m_{32} + (Z-Z_0)m_{33}} \\ y - y_0 &= \frac{-C_y\left((X-X_0)m_{21} + (Y-Y_0)m_{22} + (Z-Z_0)m_{23}\right)}{(X-X_0)m_{31} + (Y-Y_0)m_{32} + (Z-Z_0)m_{33}} \end{aligned} \quad (2)$$

where the rotation coefficients, m_{11} through m_{33}, are given by

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} \sin\phi\sin\psi\sin\theta + \cos\psi\cos\theta & \sin\phi\sin\psi\cos\theta - \cos\psi\sin\theta & \cos\phi\sin\psi \\ \cos\phi\sin\theta & \cos\phi\cos\theta & -\sin\phi \\ \sin\phi\cos\psi\sin\theta - \sin\psi\cos\theta & \sin\phi\cos\psi\cos\theta + \sin\psi\sin\theta & \cos\phi\cos\psi \end{bmatrix} \quad (3)$$

This is not a computationally efficient form of the equations. Instead of using equation (2), the camera parameters can be converted into 11 linear parameters. This is called the Direct Linear Transform (DLT) technique. To convert the physical camera parameters, let

$$P = -m_{31}X_0 - m_{32}Y_0 - m_{33}Z_0 \tag{5}$$

then the DLT parameters are

$$\begin{aligned}
PL_1 &= x_0 m_{31} - C_x m_{11} \\
PL_2 &= x_0 m_{32} - C_x m_{12} \\
PL_3 &= x_0 m_{33} - C_x m_{13} \\
PL_4 &= (C_x m_{11} - x_0 m_{31})X_0 + (C_x m_{12} - x_0 m_{32})Y_0 + (C_x m_{13} - x_0 m_{33})Z_0 \\
PL_5 &= y_0 m_{31} - C_y m_{21} \\
PL_6 &= y_0 m_{32} - C_y m_{22} \\
PL_7 &= y_0 m_{33} - C_y m_{23} \\
PL_8 &= (C_y m_{21} - y_0 m_{31})X_0 + (C_y m_{22} - y_0 m_{32})Y_0 + (C_y m_{23} - y_0 m_{33})Z_0 \\
PL_9 &= m_{31} \\
PL_{10} &= m_{32} \\
PL_{11} &= m_{33}
\end{aligned} \tag{6}$$

Now, equation (2) can be rewritten as

$$\begin{aligned}
x &= \frac{L_1 X + L_2 Y + L_3 Z + L_4}{L_9 X + L_{10} Y + L_{11} Z + 1} \\
y &= \frac{L_5 X + L_6 Y + L_7 Z + L_8}{L_9 X + L_{10} Y + L_{11} Z + 1}
\end{aligned} \tag{7}$$

Raytracing

Each pixel in an image is calculated by tracing a ray from the camera's location (X_0, Y_0, and Z_0) through the "film plane" of the image. The direction of the ray for a pixel located at x, y, is determined by the vector

$$\begin{bmatrix}
\dfrac{-x(L_{10}(L_7 + L_8) - L_6(L_{11} + 1)) + y(L_{10}(L_3 + L_4) - L_2(L_{11} + 1)) + L_2(L_7 + L_8) - L_6(L_3 + L_4)}{x(L_{10}L_5 - L_6L_9) + y(L_2L_9 - L_1L_{10}) + L_1L_6 - L_2L_5} \\
\dfrac{x(L_9(L_7 + L_8) - L_5(L_{11} + 1)) - y(L_9(L_3 + L_4) - L_1(L_{11} + 1)) - L_1(L_7 + L_8) + L_5(L_3 + L_4)}{x(L_{10}L_5 - L_6L_9) + y(L_2L_9 - L_1L_{10}) + L_1L_6 - L_2L_5} \\
1
\end{bmatrix} \tag{8}$$

The first two components are equations (7) solved for X and Y with $Z = 1$.

Only the region of space which is close to the orbital needs to be considered. As such, the ray is immediately traced to a bounding sphere which surrounds the orbital. The radius of the sphere is computed such that it contains all of the orbital's surface. If asymptotes are drawn, or a probability density plot is produced, the maximum value of Ψ^2 in the orbital is computed. The sphere which encloses the surface with 1/1000th of this probability is used.

The intersection of a ray with a sphere is given by the equation

$$x + \frac{\vec{v}}{|\vec{v}|}\left(d\cos\theta - \sqrt{r^2 - d^2\sin^2\theta}\right) \tag{9}$$

where x is the ray starting location, $\vec{v}$ is the ray's direction, θ is the angle formed by the line connecting x to the center of the sphere and $\vec{v}$, and r is the radius of the sphere. See Figure 38. If the value under the radical is negative, the ray misses the sphere.

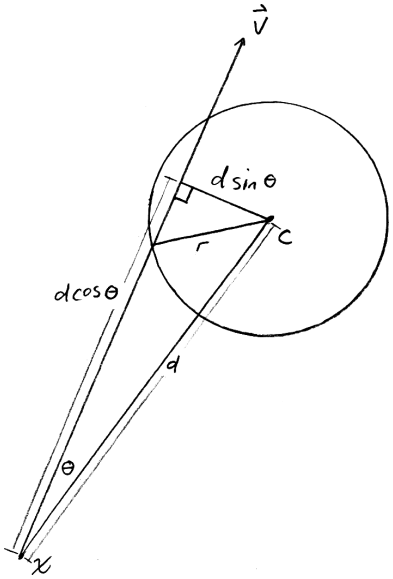


Figure 38: Intersection of a ray with a sphere

For orbitals which are drawn as completely opaque surfaces of constant probability, a modified Runge-Kutta technique is used to roughly locate the surface. this technique is modified by having a minimum and maximum positional change based on the distance from the viewpoint and the size of the orbital. After the surface has been roughly located, it is precisely located using a Newton-Rhapson process.

For orbitals which contain asymptotes, probability density plots, or partially translucent surfaces, the ray is traversed in a series of steps, with the probability calculated at each step. If the specified index of refraction is not unity, then they direction of the ray is modified according to the following equation

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \tag{10}$$

where n_1 is the index of refraction on one side of the surface, n_2 is the index of refraction on the other side, and θ_1 and θ_2 are the angles between the surface normal, as calculated from the orbital function gradient, and the ray. In the event that this results in an impossible output angle, the reflection is used instead. This is the only place in the orbital calculations where reflection is used.

GRAND TABLE

Due to file size considerations, the Grand Table is not present in this document. It can be found on the web at http://www.orbitals.com/orb/orbtable.htm. It may also be available as a separate PDF file by the name of GRANDTBL.PDF.

PROGRAM HISTORY

The Orbital Viewer program dates back many years. The following is a rough history:

1986-87: A primitive point probability program was written for the Apple II series of computers. This included a fixed animation (no user control) spinning in real-time on the Apple II plus (admittedly, it was only about 80 pixels square with a few hundred points). This version only could deal with orbitals of $n \leq 4$. This was used as a demonstration for a high-school chemistry course I was taking at the time. Programming was done in basic and assembly language.

1988-89: A version which allowed any orbital with $m = 0$ was written. This produced a point surface plot (like the polygon mode with points showing). A series of fixed animations were created for the Apple IIe computer (this time at the full resolution of 280x192). Programming was exclusively in assembly language.

1990-92: A preliminary polygon version was written for the Apple IIgs using a 3D graphics library the author also created. This was much, much too slow. At the same time, a computationally inefficient raytracing version was being run on an IBM mainframe. Programming was in assembly language, Pascal, and C.

1993-96: A very cryptic, but powerful, command line version of the raytracing method was programmed for Intel machines. This was used extensively to produce MPEG animations. Programming was in C with the actual computational aspects in very finely-tuned assembly language. An ANSI-C compatible version was also programmed.

1997-98: Finally a user interface for viewing orbitals. This program is written in C with some assembly language. An ANSI-C command line version exists, and will be proliferated to many platforms.

AUTHOR'S NOTE

It has been my pleasure to program Orbital Viewer. I hope that you enjoy using it, and I greatly value any feedback.

Contact Information

Please direct all comments, suggestions, and complaints to:

David Manthey
P. O. Box 27
Troy, New York 12181-0027
manthey@orbitals.com

Acknowledgements

Although all of the computer code and documentation have been written by myself, I have several people to thank:

William Manthey, my father, for getting me into this. I first wrote a simple program to generate point density plots for a high school chemistry course, partly his suggestion. Since then, orbitals have become a major hobby, as this document and the program itself testifies.

Catherine Manthey, my mother, for the Orbital Quilt. See Figure 39 and Figure 40.

Figure 39: The Orbital Quilt, made by Catherine Manthey

Figure 40: Underside detail on the Orbital Quilt

Benjamin Manthey, my older brother, for his substantial help in beta testing the program. Without his assistance, the Orbital Viewer program would be but a fragment of its present self.

REFERENCES

Abdel-Aziz, Y. I. and H. M. Karara, "Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close Range Photogrammetry," *Proceedings of ASP/UI Symposium on Close-Range Photogrammetry*, January 1971.

Ballhausen, Carl J., *Introduction to Ligand Field Theory*, New York: McGraw-Hill Book Company, Inc., 1962.

Carneiro, Bernardo Piquet, Claudio T. Silva, and Arie E. Kaufman, "Tetra-Cubes: An algorithm to generate 3D isosurfaces based upon tetrahedra", *SIGGRAPH '96 Proceedings*, Vol. 30 (1996), pp. 205-210.

Condon, E. U., and G. H. Shortley, *The Theory of Atomic Spectra*, Cambridge: University Press, 1959.

King, Gerald W., *Spectroscopy and Molecular Structure*, New York: Holt, Rinehart and Winston, Inc., 1965.

Lorensen, William E. and Harvey E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *Computer Graphics (SIGGRAPH '87 Proceedings)*, Vol. 21 (July 1987), pp. 163–169.

Margolis, Emil J., *Bonding and Structure*, New York: Meredith Corporation, 1968.

Offenhartz, Peter O'D., *Atomic and Molecular Orbital Theory*, New York: McGraw-Hill Book Company, Inc., 1970.

Shih, Tian-Yuan, "The Reversibility of Six Geometric Color Spaces", *Photogrammetric Engineering and Remote Sensing*, Vol. 61, No. 10 (October 1995), pp. 1223-1232.

Toutin, Theirry and Benoit Rivard, "A New Tool for Depth Perception of Multi-Source Data", *Photogrammetric Engineering and Remote Sensing*, Vol. 61, No. 10 (October 1995), pp. 1209-1211.