Jmol 11.1.5 Perspective/Navigation Model (rev.07.01.05)

Bob Hanson, hansonr@stolaf.edu
1/5/2007

Jmol 11.1 introduces the possibility of scripted navigation *through* a model. In order to achieve this effect, changes were necessary to the underlying perspective model of Jmol 10.2/11.0. For backward compatibility, the default perspective model in Jmol 11.1 is still that of 10.2, but issuing of either of the following commands initiates use of the more flexible 11.1 model. "Jmol" in discussions below refer to Jmol 11.1 specifically. The 10.2 model was written by Miguel Howard; the 11.1 model was written by Bob Hanson.

```
set navigationMode
set perspectiveModel 11
```

Jmol Coordinate Transformation

Jmol applies perspective as the final step in the transformation of 3D molecular coordinates (xyz) to 2D screen coordinates (XY) and an associated standard z-buffer value (Z, with 1 being nearest the observer and infinity being far away from the observer). All coordinates in Jmol are Java float values, but there are times when integers are used instead. In discussions below, (m) or (xyz) indicates molecular coordinates; (s) or (XYZ) refers to screen coordinates (0,0 in upper left corner, Z increasing away from the user); and (p) refers to vertical plane number, which is linearly related to Z as described below.

The overall transformation is carried out in two steps, one involving a matrix, *matrixTransform,* and one involving a function, *getPerspectiveFactor()* according to the following steps:

1a) translate the fixed rotation center to (0,0,0) (m)
1b) apply the current molecular rotation (m)
1c) apply zoom-based scaling, *scalePixelsPerAngstrom* (s)
1d) add a Z-translation, *modelCenterOffset* (s)

2a) if a Z value is undefined or less than 1, set it equal to 1
2b) translate the center of perspective to (0,0) (s)
2c) apply a perspective factor to X and Y based on the value of Z. For low to moderate Z values, this is just:

X *= *referencePlaneOffset / Z;*
Y *= *referencePlaneOffset / Z;*

(For extraordinarily high zoom levels, there may be some additional scaling.)
2d) move the center of perspective to the appropriate XY screen position.

The *referencePlaneOffset* parameter is constant. Thus, the perspective "function" is contained in the value of Z. Its calculation is the subject of this summary.

Zoom is taken into account as a factor in *scalePixelsPerAngstrom*. (In Jmol 10.2 this quantity was also dependent upon the "camera depth", but that association has been removed in Jmol 11.1.)

Because Jmol must scale properly for different screen sizes, it is convenient to define measurements in multiples of *screenPixelCounts*. The Jmol model defines the *screenPixelCount* as two less than either the height or the width of the screen in pixels, determined as follows: If the default *set zoomLarge* is in effect, then the larger dimension of height and width is used; if the user/developer has issued *set zoomLarge FALSE*, then the smaller of the two is used. Thus, for a 400-wide by 300-high applet, by default *screenPixelCount* = 398. The designation (p) for units in the following discussion implies that the quantity is measured by taking screen pixels and dividing by *screenPixelCount*, thus making it general to any screen size.

The Jmol 11 Perspective Model

The Jmol perspective model is most easily described in terms of a perspective diagram (Figure 1) in which the vertical axis represents 1/*f*, where *f* is the perspective factor applied to a given point (*referencePlaneOffset / Z)*, and the horizontal axis is an X or Y screen coordinate. The horizontal center of the diagram is 0, the center of perspective.

The easiest way to think about it is that you are omnisciently looking DOWN from the top of the model. The horizontal distance between the diagonal lines in all cases represents the actual screen width – items near the rear will be compressed to fit this number of pixels; items near the front will be stretched. This produces the illusion of depth.

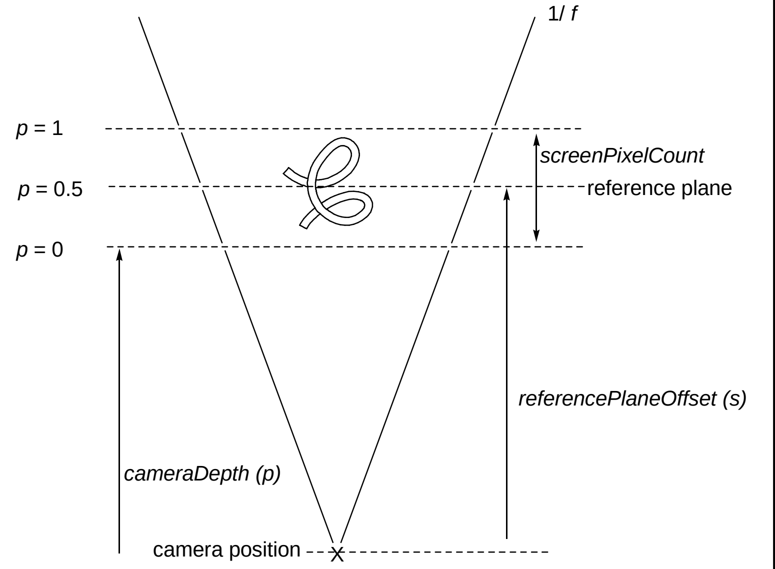

Figure 1. Jmol linear perspective model. 1/*f* increases with distance from the user.

In Jmol, the function *f* is given by:

$$f = \frac{referencePlaneOffset}{Z} = \frac{c + 0.5}{c + p}$$

where *c* is the “*cameraDepth*” (p), and *p* is the Z-position of the point relative to a 0-plane situated just in front of the model when the default zoom of 100% is applied. Note that by this definition, $1/f$ is linearly related to *p*, increasing with increasing distance from the user; thus the linear V-shaped diagram. Note that the quantity “$c + 0.5$” is a measure of the distance from the camera to a reference plane at the center of the model, where *p* is 0.5. The perspective factor at this position is 1; a flat model will appear the same in Jmol with or without perspective set (using *set perspectiveDepth TRUE/FALSE*). The default value for *cameraDepth* in Jmol is 3.0, but much smaller camera depths, around 0.5, can be used for dramatic effects. This value can be set using *set cameraDepth x.x*, where *x.x* is the desired depth in multiples of *screenPixelCount.*

Zoom

Zoom when NOT navigating in Jmol 11 is simply applied by keeping the center of the model in place and making the model larger (Figure 2). The default 100% zoom setting determines the default scale based on the nominal *modelRadius (m)* of the model. This is the smallest radius in molecular coordinates that perfectly contains the entire model. The idea is simply that for any given model, "*zoom 100*" should display the model in such a way that it is as big as it can be without overflowing the screen no matter how the user rotates it. Thus we have in general:

scalePixelsPerAngstrom
 = *zoom* * *screenPixelCount* /
 (2 * *modelRadius*)

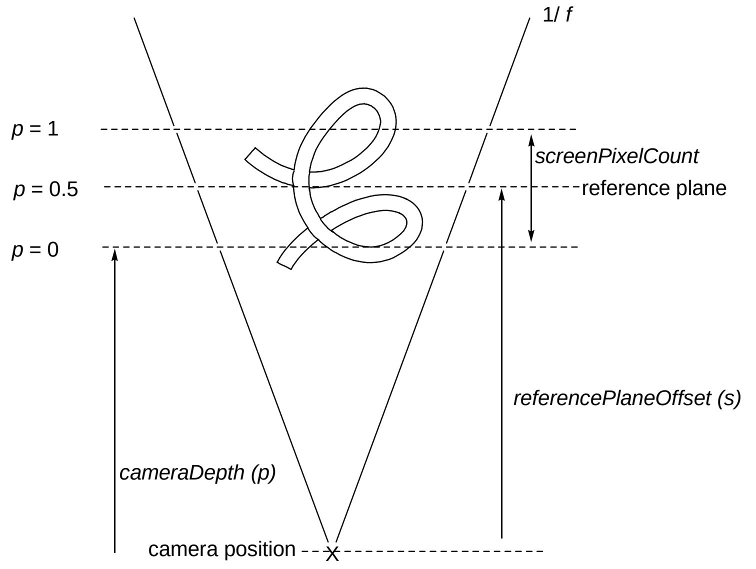

Figure 2. With *zoomPercent = 200*, the model has the same center but is twice as large in all dimensions.

Navigation

The above simple model handles most needs for perspective. Late in 2006 Charles Xie, working on a project with the Concord Consortium, suggested that Jmol be extended in such a way as to allow more dramatic "fly-throughs" of molecular systems. One might, for example, navigate through a nanotube, zip through the core of an alpha helix, or ride the wave of a beta-pleated sheet. The result of that collaboration is given below. It involves the addition of just three additional components to the model (Figure 3), namely:

visualRange	defines the Z-position of the observer
navigationOffset	defines the XY-position of the observer
navigationDepth	defines the molecular point of the observer

visualRange (m) A parameter defining the minimum distance across the screen, in Angstroms, for objects in a given Z-plane to be displayed. The idea is that the difference between navigating through a molecule and looking at it through a telescope is that when navigating you are in front of part of the model, while with a telescope there would be problems with atoms very "near" the user eclipsing others further away. We need navigation, not telescoping.

navigationOffset (s) The position in screen (XYZ) coordinates for the point in space corresponding to the user. The *Z* position of the *navigationOffset* point is in the plane defined by *visualRange*.

navigationDepth In order to give the appearance of the model rotating around the observer even though it is really being transformed in relation to a fixed rotation center, the navigation point can be defined in relation to the overall depth of the model.

In typical navigation, the only actual change that occurs is in the position and orientation of the model itself. In this scheme, the position of the *visualRange* plane and thus the *Z* position of the *navigationOffset* do not change. This value is set at *p* = 0.5. As the user navigates “into” the model, the model simply moves forward – the *navigationDepth* decreases, and the underlying molecular coordinates corresponding to the *navigationOffset* are continually adjusted. As the user turns left or right or pitches upward or downward, the model simply rotates and translates appropriately.

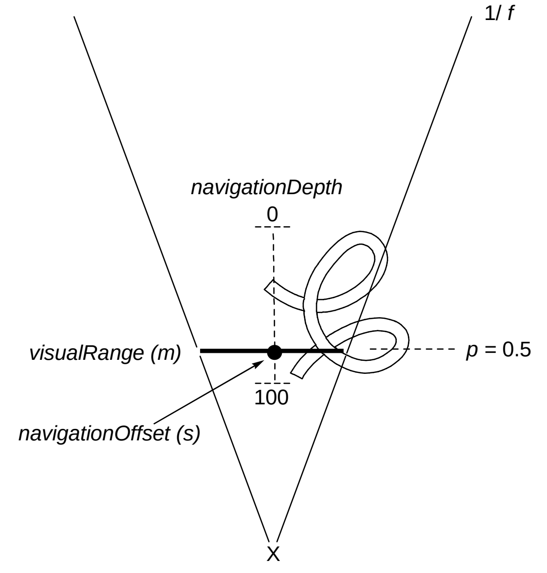

Figure 3. The Jmol navigation model includes a clipping plane defined by *visualRange (m)*, an observer-based *navigationOffset (s)*, and an adjustable *navigationDepth (s).*

Additional Jmol Navigation Parameters

While only these three independent parameters are necessary, Jmol uses several additional parameters for convenience only (Figure 4). These include:

modelCenterOffset, the XYZ position of the fixed rotation center in relation to the camera.

navigationCenter tracks the underlying molecular *xyz* position of the observer.

When the user scans to the right, the model in response rotates clockwise (as seen from the top), and translations are applied to keep the *navigationCenter* at its current *referencePlaneOffset* position. In addition, the user can control the XY coordinates of the *navigationOffset* by pressing SHIFT while using the arrow keys.

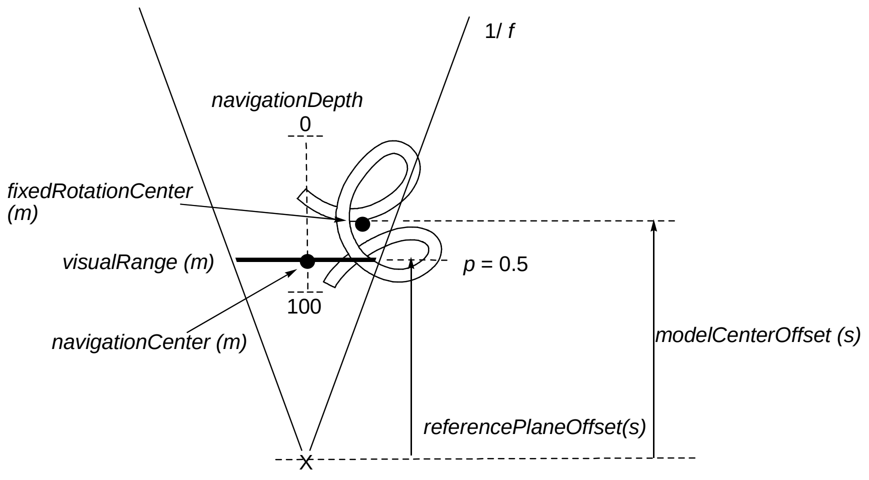

Figure 4. The Jmol navigation model includes a clipping plane defined by *visualRange (m)*, an observer-based *navigationCenter (m)*, and an adjustable *navigationDepth (s).*

Zoom and Distortion In Relation to Navigation

Inspection of Figure 4 suggests that, given the above measures, zoom is no longer an independent variable. This allows Jmol navigation mode to dispense with zoom. “Zooming” in navigation mode simply amounts to moving a standard-scale (*zoom 100*) model forward. Basically, we have a simple linear relationship between apparent zoom and *modelCenterOffset.* While there is always an equivalent *zoomPercent* corresponding to any particular observer position, Jmol does not actually use that zoom measure in any calculation when in navigation mode. The relationship is simply:

*modelCenterOffset = offset100 * (100 / zoomPercent)*

where *offset100* is the offset of the model center when it is to appear at a zoom of 100:

*offset100 = (2 * modelRadius) / visualRange * referencePlaneOffset*

An interesting problem with zoom arises within this navigation model. When *modelCenterOffset* is zero, and the *fixedRotationCenter* coincides with the camera position, then *zoomPercent* is undefined, and when *modelCenterOffset* is less than zero we have the equivalent of negative zoom. (It is interesting that the Jmol 10 perspective model sometimes required enormous amounts of zooming – up to 200000 *percent* – primarily because in that model there was no option to bring the object center closer to the camera. Basically what was happening was that as the model was magnified its rear portion became more distant from the observer. What we were seeing was an approximation of the case in navigation when the model center perfectly coincides with the camera position, and magnification of the model simply has no effect.

The result of this analysis is that navigation mode provides access to otherwise inaccessible “negative” and “undefined” zoom levels. The practical result is that while the *show zoom* command in Jmol 11 will give a measure of the equivalent *zoomPercent* while in navigation mode, this quantity may be negative or (effectively) infinite. Jmol is programmed to not allow exiting of navigation mode unless an equivalent non-navigation mode rendering can be produced. This requires *zoomPercent* > 5.

Finally, there is one subtle hitch: In molecular visualization it is often important to portray spheres at different differences. Since the illusion of perspective is created by stretching the X and Y directions specifically, the model no longer properly depicts Z separation at the same scale as X/Y dimension. In effect we do not have an affine transformation (a transformation that preserves distances and linear relationships). This generally shouldn’t be a problem, because we cannot see the Z direction anyway, but when clipping is involved, as with navigation, objects such as spheres that are depicted based on their center position and radius (and are not actually distorted) may appear in front of other objects when they really are not. By clipping at the navigation point situated in the *referenceOffsetPlane*, where X, Y, and Z are all properly dimensioned, we avoid this problem in Jmol.

Scripted Navigation

Navigation can be scripted. Commands include:

```
set cameraDepth x.xx
set hideNavigationPoint
set navigationMode TRUE/FALSE
set navigationSlab [depth from navigation point; positive toward user]
set navigationSpeed [5]
set navigationPeriodic # for crystals, creates the effect of an infinite array in all directions
set perspectiveModel 11/10 # (10 disallows navigation)
set picking navigation
set rotationRadius x.xx
set showNavigationPointAlways
set visualRange x.xx

navigation nSec center {x y z}
navigation nSec center $object
navigation nSec center (atom expression)
navigation nSec depth p # a depth value, like slab, in percent (0 rear, 100 front)
navigation nSec path $object indexStart indexEnd
navigation nSec path (atom expr) {x y z} (atomexpr) (atomexpr) {x y z} etc...
navigation nSec path {x y z theta} {x y z theta}{x y z theta}{x y z theta}...
navigation nSec rotate X degrees
navigation nSec rotate Y degrees
navigation nSec rotate Z degrees
navigation nSec trace (atom expression)
navigation nSec translate x.xx y.yy  # percentages; 0 0 center
navigation nSec translate X x.xx # relative percent
navigation nSec translate Y y.yy # relative percent
navigation nSec translate {x y z}
navigation nSec translate $object # could be a draw object
navigation nSec translate (atom expression) #average of values
```

The nSec parameter is optional and defaults to 2 seconds. Commands can be stacked together for convenience only using “/”: `navigation depth 50 / rotate X 30 / depth 20 / translate X 10`

Note that “depth 20” after the rotation is different from “depth 50” before – it is along a different axis. Also note that concurrent spinning of the model, though possible, is not recommended during navigation. Many of these commands depend upon the current model orientation. If that is changed by the user or by spinning during navigation, the end result is not guaranteed. In principal one could safeguard against this, but we are not there yet.

In addition, an extension to the moveTo command allows simultaneous reorienting and navigation:

```
moveTo [timeSec] {x y z w} zoom [xTrans yTrans]
                            ({center}) [modelRadius] ({navCenter}) [navX navY navDepth]
```

The center and navCenter parameters can be any one of (a) a draw object preceded by $, (b) an atom expression in parentheses, or (c) a (possibly fractional) coordinate in braces.

Additional commands that can be used to help develop pages using navigation include:

```
show/save/restore state
show/save/restore orientation
show set  # displays all settings
```