Pablo User Guide

By Gregg Tracton
Pablo Version: 2001-08-01
Last Update to document: 8/3/01 1:32 PM

To view this document requires a HTML Browser capable of	HTML level 2, simple tables, color, images, no frames
To Print document, use	Adobe Acrobat version 4 or above

Pablo changes everyday. This document is a snapshot of the current version.

Pablo Programmers:

Tom Fletcher	fletcher@cs.unc.edu
Yonatan "Yoni" Fridman	fridman@cs.unc.edu
Graham Gash	gash@cs.unc.edu
Sarang Joshi	joshi@radonc.unc.edu
Martin Styner	styner@cs.unc.edu
Andrew Thall	thall@cs.unc.edu
Gregg Tracton	tracton@radonc.unc.edu
Paul Yushkevitz	pauly@cs.unc.edu
unnamed others...	gurus@anonymous.net ☺

1 Table of Contents

You can click on a section to speed you to that section.

2 Scope

2.1 Audience: Shape Researchers:

- do not need to know how to program,
- must know how to edit structured text files,
- should know 3D graphics and geometry terms (such as camera, view, surface, cut plane, isotropic, coordinate systems), and
- should be familiar with medical anatomy.

2.2 Related Documents:

- "Deformable M-Reps for 3D Medical Image Segmentation", Pizer et al, submitted to MedIA August 2000. (on defmreps web page)
- "***Pablo*** Tutorial on Crafting Shape Models", Gregg Tracton.
- "***Pablo*** Installation Guide", Tom Fletcher.
- **defmreps** web page contains links to all things m-reps: papers, segmentations results movies, Radiotherapy uses, statistical shape characterization for Neuroscience, etc. See http://www.cs.unc.edu/Research/Image/MIDAG/defmreps

3 Purpose: What is Pablo, Who Uses It

Pablo is research software used to position and shape m-rep models, possibly in the context of a 3D 12-bit grayscale image and/or 3D triangular tiles to approximate the segmentation being modeled. It is also a good 3D viewer of these structures; displays include a 3D perspective window, multiple 2D cut plane windows (anchored to the image's plane or anchored to one or more atoms), and a 2D ribbon view through multiple atoms.

At any one time, pablo can know about:

- a single model of multiple figures which represents one or more objects such as anatomy, surgical clips, tumors, etc
- a single target image and a single training image, both 3D
- a single tile set.

It can perform operations on arbitrary sets of atoms. It can interpolate between the atoms in a figure to compute both smooth medial sheets (the center of the object used by model builders) and the middle of the implied boundary (the surface of the object, estimated at some scale). It can fit a model to the underlying image in many ways: manual (under user control), whole-model (using a similarity transform over all figures), per object (geometric constraints), per figure (similarity transform), atom deformation (per atom), [LATER: and boundary displacement (per boundary position).] It can write an image representing the modeled object, for instance, with 1 pixel values inside the object and 0 pixels values elsewhere, [TECH: or intensities near the boundary masks which indicate values from the training image].

We anticipate two distinct classes of users: **model builders** and **model fitters**. Fitters adjust pre-built models to a particular image. Our plan is that they should only need to know about figures and surfaces. Builders need to know what fitters know, plus have a working knowledge of the medial sheet of all the models in a anatomical site (in the current user interface), how figures fit together to form objects, and the intimate details of atoms.

Pablo runs on Microsoft Windows, expects a 3-button mouse and uses advanced openGL 1.1 features of 3D graphics boards. A 2-button mouse can emulate a 3-button mouse using the Shift key to indicate the missing middle mouse button, useful for laptops having only 2-button capability. [TECH: A source-code compatible Solaris/SGI version is in the works and we always write our programs with the expectation of multiple OS environments.]

4 Nomenclature: Fonts, Colors, Styles in this Guide

Styles, fonts and colors may have specific meanings.

Underline	An underlined word indicates emphasis.
Bold	A word in the bold text attribute delimits: • words you'll see, literally, in the user interface or document • words that introduce a topic
italic	this text attribute defines proper words with meanings specific to this design. Most browsers draw emphasized text in an italic font.
Colors	See context. Usually indicates a color in the user interface or document. Blue text means the section is out of date.
`primitive[0][0]`	this font indicates something you type.
[TBD]	means "To Be Defined"
[LATER: text**]**	describes features that are planned
[TECH: text**]**	technical text intended for programmers, advanced users or model builders.
[WINDOWS: text**]**	describes features that are implemented for Microsoft Windows platforms only.
[0..2]	the closed range of numbers from 0 through 2, including both 0 and 2. This range can be either real numbers or integers, distinguishable from context.
20000614	versions are defined by dates in YYYYMMDD format. This sample means June 14, 2000, the first time m-reps worked on a real clinical image.
File -> Load Model...	means left-click on the **File** menu to expose the sub-menu, then select the **Load Model...** menu option. May also refer to a notebook tabbed window and it's tabs. **...** means that a dialog box will pop-up to collect more details before performing the action.
Cntl+z	is an "accelerator" - a keyboard equivalent for an action. For this example, hold down the **control** key and press the **z** key, then release both.

5 Concepts: Terms You Need to Know

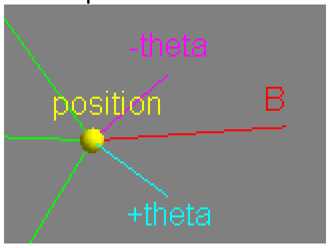
A user needs to understand these concepts for effective use.

5.1 M-rep model

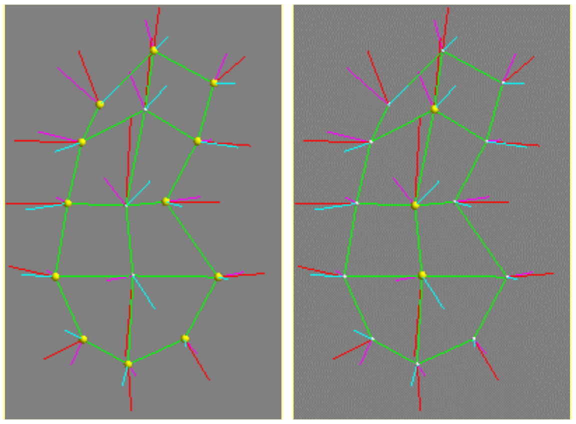
A model, as manipulated by this program, is a set of ***objects*** containing ***figures***, some per-figure attributes (such as polarity, display color or name) and relationship information between objects and/or figures that determine their properties during optimization. A figure represents a significant portion of an object, such as a finger of a hand or, to illustrate negative figures, a hole in a bone. A model is contained entirely in a single text file called a ***m3d*** file.

5.2 Atom, Primitive

An atom (previously called a "primitive") determines the shape of two related patches of the object's surface and the contained 3-D space in-between. Some atoms' patches wrap around the edge of the object and so the surface portions are joined to form a single sheet. Each patch is usually small and is measured at a specific scale. [TECH: an atom has a **position** somewhere near the object's medial axis (a curvy plane in the object's center), an expandable **B vector** (Bisector) indicating the main direction of the narrowing of the figure, two Y vectors at **+theta** (called Y0) and **-theta** (Y1) degrees from the B vector that imply the object's boundary, an N vector (not shown) perpendicular to the B vector and normal to the medial plane, a B-Perp vector (not shown) orthogonal to the B and N vectors, a width indicating the distance at which an implied boundary is most probable:



A figure's contains a NxM mesh of atoms, where N = 2^n+1 and M = 2^m+1 is suggested for ease of subdivision. Typical values are 3, 5, 9 (7 is not suggested). In this 3x5 mesh

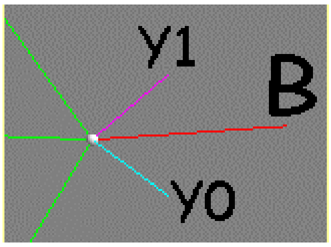


End Atoms in yellow | Standard Atoms in yellow

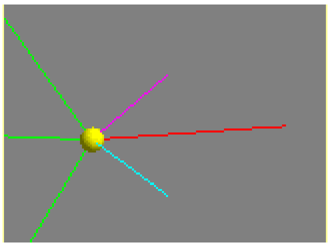
we see that each ***end*** atom has 3 neighboring atoms, indicated by the **green lines**, and that each ***standard*** type has 4 neighbors.]

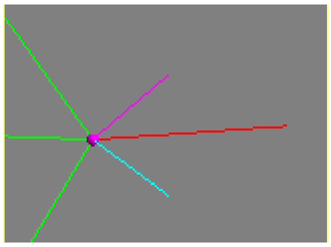
5.3 Selecting & Marking Atoms

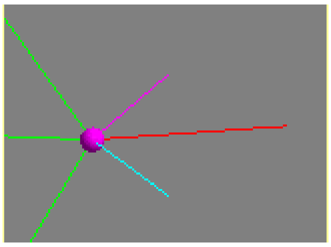
Atoms change color or size when **selected** or **marked** for manipulation. An unselected atom's center is a small white ball. The ball is connected to neighboring mesh atoms by green lines (color is user-chosen). When selected, large balls are drawn. When marked, magenta balls are drawn. One (only one) atom may be **marked** at at time.



Normal


Selected


Marked


Marked & Selected

You can't select/mark atoms that you can't see (behind opaque tiles). See Mouse Buttons.

5.4 Image

A 3D grid of voxels each containing a 16-bit signed scalar and occupying a parallelpiped ("3D rectilinear field") of space. [TECH: Currently, the grid must be regularly spaced in each of it's X, Y and Z dimensions, but these spacings need not be equal (X is commonly the same as Y, however). The Z dimension is across the natural "slice" orientation of the image acquisition, but it is not defined as to how this correlates with true space in the real world (for instance, it could be rotated 90 degrees). The file formats are ***raw3*** and ***UNC metaheader***, but irregular Z spacings may be required at some point and so these will be expanded.]

5.4.1 Unit Grayscale/Color Space

Grayscale images may have up to 65,000 distinct grayscale pixel values. [TECH: For grayscale images, these are mapped to the float range [0..1] representing the values seen in that image. The image contains a header telling the possible range.]
Color images are used for display only and each pixel contains three real numbers [0..1] representing Red, Green and Blue color components which are additively mixed.

5.5 Tiles

An arbitrary set of 3D triangles representing a 3D surface. [TECH: The normals all face toward the outside of the "object" being defined. Note that holes and multiple surfaces are allowed and the direction the normal faces is defined in any way the tile creator desires in these cases. We have not defined what "outside" means in a negative figure yet.]

5.6 Unit Coordinate System

Models, images and tiles are all scaled to the same coordinate system which is defined as a isotropic unit cube in 3D with each dimension's domain [0..1]. The cube does not have to be completely filled. [TECH: if the image's space is not cubic then 1 or 2 of the dimensions will occupy [0..N] where N is somewhere in (0..1] and is the ratio of the smaller dimension to the larger dimension.]

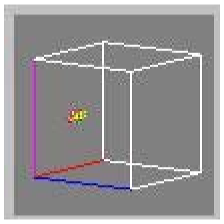
5.7 Object Hierarchy

This gives you a clue of what sort of information should be in each file and the attributes the user interface may be able to edit and/or display. An example of how to read this is "a model contains a set of figures and a set of figureTrees; each figure contains a figure ID, a color, ...":

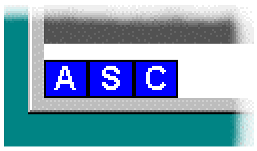
- Model
  - Figures (set)
    - Figure ID: [0..N]
    - Figure's Name (user assigned)
    - display color: Red, Green, Blue
    - positivePolarity: 1 for positive, 0 for negative
    - positiveSpace: 1 for protrusion figure, 0 for indentation figure
    - type: QuadFigure
    - numRows
    - numColumns
    - atoms (Mesh)
      - r: length of Y vectors
      - elongation: [1..infinity]
      - orientation: qx, qy, qz, qx
      - selection flag
      - mark flag
      - theta angle: degrees
      - type: EndPrimitive/StandardPrimitive
      - position: x, y, z
  - FigureTrees (set)
    - parent figure ID
    - childLinks (set)
      - blendAmount: [0..1]
      - blendExtent: [0..1]
      - child figure ID
      - childLinks (set)
        - child atom ID
        - (u,v,t) coordinate of parent
- Image
  - textual description
  - voxel dimensions, eg, 256x256x62
  - voxel size, eg, .1x.1x.4 cm
  - voxel orientation: can invert direction of axes
  - voxel values
- TileSet
  - Tiles (set): x1,y1,z1, x2,y2,z2, x3,y3,z3

6 Conventions: What Pablo Assumes

- Reads these file formats:
  - images: ***raw3*** or ***mha*** (Aylward Metaheader) images, which the user generates,
  - tiles: ***UNC tile*** format, which the user generates,
  - models: ***m3d*** m-rep format, or unsupported ***mod*** format
  - Tiles and models are encoded in ***PaulY*** format (see Editing Model Files), which is easily read and changed by users with any text editor program.
- Writes these file formats:
  - ***raw3*** images
  - ***m3d*** m-rep format.
- To make tiles, use one of your own tools or a combination of the MASK contouring tool or any other contouring tool and the CTI contour-to-tile tool. (Both from http://www.radonc.unc.edu/tools). We find it more convenient to create tiles for each figure and then simply concatenate them together to form a single file containing tiles for the whole object (or any subset of figures), when needed, but smooth corners might not be represented accurately that way.
- Load the image before the tiles or model, because the image contains values from which is derived a scale which defines the size of a pixel (in centimeters). This allows one to use a model across multiple images with different pixel sizes and was chosen instead of cm as the coordinate system because a model ***has no actual physical size***, that is, coordinates are flexible enough to be interpreted as cm, inches, leagues, or even light years. The tiles are not in the correct position until a corresponding image has been loaded.
- Axes are color coded. X,Y,Z are **Red**, **Magenta**, **Blue**. The X & Y axes are considered "in plane" w.r.t. the image; the Z axis is "cross plane," by convention. Orientations are actually arbitrary in practice.



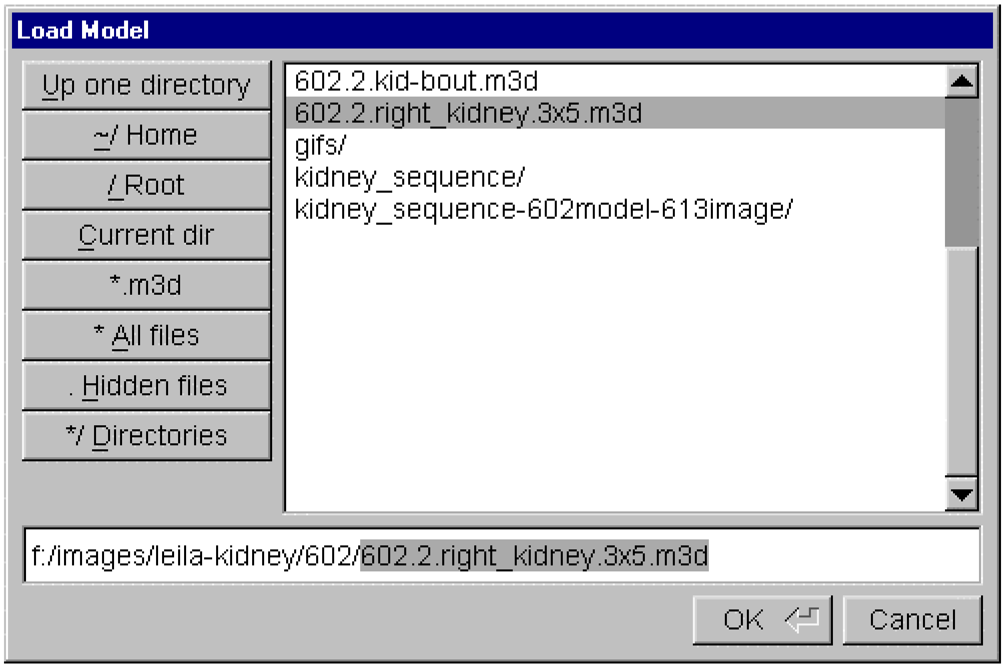
- **trackball** - the views are manipulated in 3D using a "virtual trackball." To rotate objects or cameras, imagine that on the model window is centered a 3D sphere that always touches the 4 corners of the window. The mouse click is projected up onto the sphere's surface and selects a position on the ball. Mouse movements are also projected up and the ball is rotated in 3D around it's center to maintain the selected model position under the mouse. The view and/or atoms follows the sphere. For example, to tumble the scene towards the camera, press the left mouse button at the top center of the window and drag the mouse to the bottom center. When the scene is oriented properly, release the mouse button.
  Lost? Press **Display Control -> General -> Reset View** to set the camera to it's initial pose, position and scale. Or press a button (see picture at right) to reset the camera to Anterior (view from +Z), Superior (from +Y) or Coronal (from +X). **Undo** does not affect the camera.



- **Alpha** sliders control opacity of tiles, [LATER: images and model surfaces]. Move the slider to **1.0** for opaque, **.5** for transparent, **0.0** for invisible.
- Only the model view window may be resized. Cut Plane view windows are statically calculated and so are not resizable.
- Checkboxes are enabled when they are red, disabled when gray. Click the box to enable or disable

7 File Browser (FLTK Style)

When Pablo needs you to specify the names and paths (folders) that files should be read from and written to, a file browser will appear.



602.2.kid-bout.m3d
602.2.right_kidney.3x5.m3d
gifs/
kidney_sequence/
kidney_sequence-602model-613image/

f:/images/leila-kidney/602/602.2.right_kidney.3x5.m3d

It may be different from file browsers you have seen before:

- Read the prompt in the title. Sometimes a "save model" file browser will appear when you are expecting a "load model" browser, because Pablo can only hold a single model at a time and wants to give you a chance to save changes to the last model before replacing it with the new.
- [Windows] To look at the files/folders on a different disk, replace all the text on the bottom text entry with the disk letter, eg, "f:/" to look at the F disk.
- Clearing out the text will show your current working directory.
- Files have types (eg, raw3 for images, m3d for models) as indicated by last few letters of the file's name after the last dot. Only the files of the type that the application is expecting will be shown in the browser's list initially, and only one type is specified by the program. You can force it to show all the files (including the hidden ones!) using the buttons on the left side of the browser window.
- The **Tab** key expands a partial selection. For example, if the name of a directory is "leila-kidney" you may type just `le` and then **Tab** and the rest of the directory's name will appear. It must be uniquely specified, that is, if there's another directory called "leonard-zeppelin" then "le" is not not enough letters for the browser to tell which you mean. Type more letters, then another **Tab**.
- Use single clicks to select directories. If you should use a double-click then you will select both the directory and some file in that dir, which can crash the program if the errant file is of the wrong format.
- Resize this window if the filenames are too long to be displayed fully.
- **Up** & **Down** arrow keys select the previous and next file/directory.

8 Mouse Buttons

This describes mouse actions in the 3D model view window.

- A 3-button mouse allows use of the Control and Alt "modifier" (shift) keys: hold down the modifier key and click the mouse button(s), then release the modifier key. Build into Pablo is to use a 2-button mouse by substituting Shift-left for the middle mouse button; as indicated at the bottom of the table below
- Scale-width (Alt&Shift-right) is only available by Shift'ing. We ran out of buttons…
- The *view* actions affect the camera only. The atoms are not moved relative to the axes, images, tiles or each other.
- *Select* and *mark* actions toggle atom states: if an atom had been selected before the action, the action will deselect the atom. When using the marquee, each atom is toggled independently of the others
- The marquee acts on multiple atoms at once. Drag out a rectangular region. You'll see a yellow line indicating the region. All the atoms within the rectangle will be affected, up to the limits of the graphics hardware (usually 255 atoms). If you start the drag on a atom then you'll just affect that one atom, which is probably not your intention.
- COG = Center of Gravity of all selected atoms.

shift]			
	figure, or select multiple atoms with marquee		
	rotate selected atoms with trackball	translate selected atoms in view plane (orthogonal to line of sight)	scale selected atoms around COG: drag mouse down screen to enlarge.
Shift	translate view	-	-
Shift & Alt	translate selected atoms (see Alt-middle)	-	Scale selected atoms ***and*** change their width

9 Views

9.1 Model View Window

This view shows a 3D perspective visualization of the tiles, image slices/axes and models. The user can interactively manipulate the pose/position/zoom of the camera and atoms directly in this window by translating, rotating, scaling, selecting and marking. See Mouse Buttons.

Tiles are drawn as two-sided: the inside face of each tile is a different color than the outside face for those situations in which the tiled surface folds back on itself.

If you resize the window to a non-square the scene will always remain square causing unused space at the side or bottom edge of the window.

9.2 Slice View Windows

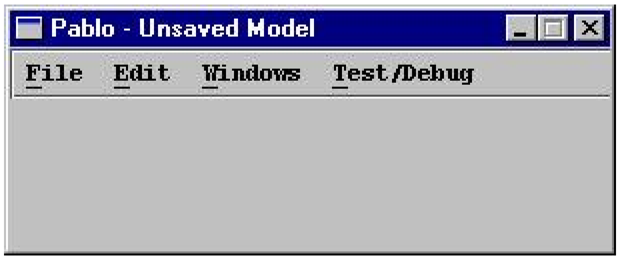
The **Main -> Windows -> Atom Cut Planes** views help to visualize the image boundaries close to a particular atom. It houses checkboxes that expose any of 6 windows, each with a 2D plane through or near the marked atom that can display an arbitrary image slice aligned with the marked atom, lines representing the atoms (with axes) in the marked atom's slice, a curve indicating the implied boundary, and the intersection of a slab aligned with the atom's geometry with the tiles. There are various orientations of this view with respect to the marked atom:

1. **Crest**: b/n plane
2. **Atom**: b/b-Perp plane
3. **BPerp-N**: n/b-Perp plane
4. **Starboard sail**: +theta/n
5. **Port sail**: -theta/n
6. **Involutes**: chord/n

[LATER: needs picture]

The **chord** is the line segment between the tips of the +theta and -theta vectors, and observant readers will note that it does not intersect the n vector in the last view. Instead, the n vector is translated so that it intersects the chord and that plane is used. The views are hard-coded.

10 Main Window

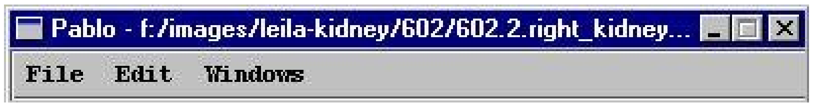


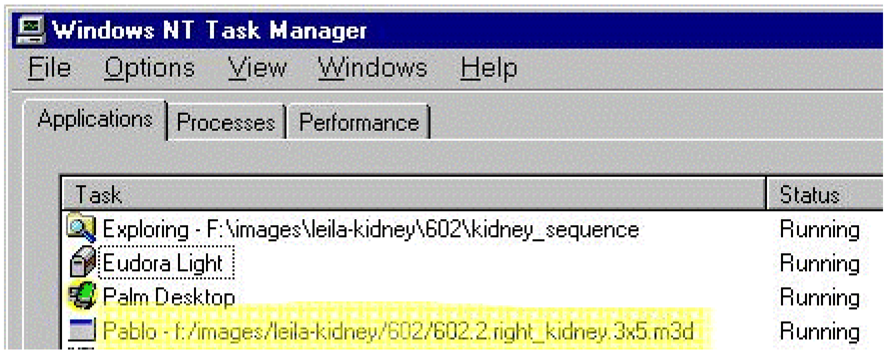
Contains top-level menus for:

- reading/writing files
- undo/redo atom modifications
- atom selection
- figure copy-and-paste
- test/debug controls
- mirror atoms around an X plane
- figure creation/destruction
- showing/hiding control/view windows: preferences, display, optimization, atom cut plane, visibility, object-object constraint

The name of the current model's file (see picture below), or **Unsaved Model** (see picture above), is displayed in the title bar of the window. If you are concerned about patient confidentiality (not disclosing your patient's name to outside parties) then you should be careful not to name the model or <u>any folder in the path to the model file</u> with the patient's name, as this may show up in window dumps useful for publishing results.

[WINDOWS NT: If you need to read the filename and it is clipped to the size of the bar, a trick is to start the Windows task manager and expand the width of the **Task** column until the filename can be read at the end of the task:

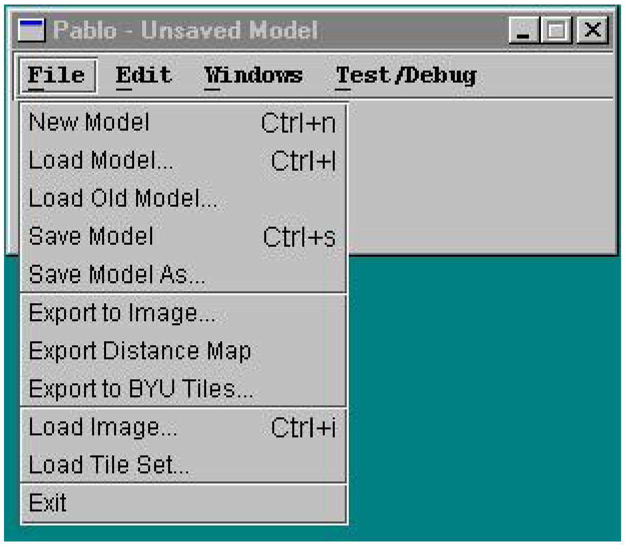




Task	Status
Exploring - F:\images\leila-kidney\602\kidney_sequence	Running
Eudora Light	Running
Palm Desktop	Running
Pablo - f:/images/leila-kidney/602/602.2.right_kidney.3x5.m3d	Running

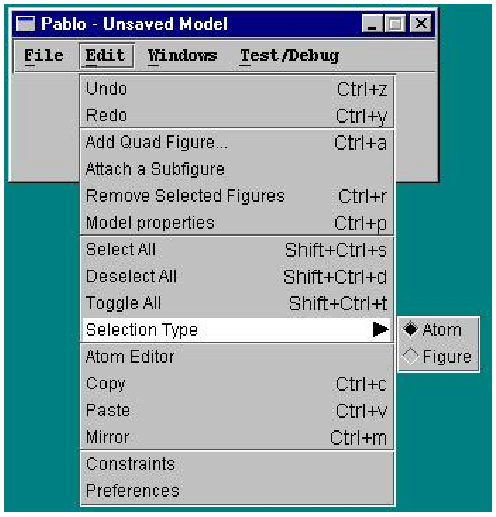
. You can also read this info in **Windows -> Display Controls -> General**.]

10.1 Main Window -> File

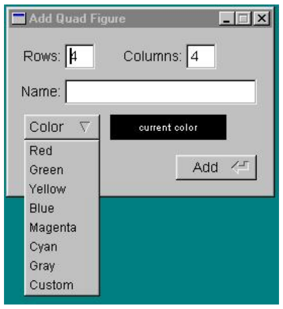


- **File -> New Model...** deletes the current model.
- **File -> Load Model...** reads a m-rep model (.m3d) from disk, replacing the current model. *Pay careful attention in case it a*sks if you want to save changes to the current model first
- **File -> Load Old Model...** is for backwards compatibility and probably will not work anymore. It's like **Load Model** except that it reads the old format (.mod). The atom types (EndPrimitive or StandardPrimitive) are calculated and the elongations are all set to 1.0.
- **File -> Save Model...** creates or overwrites (does <u>not</u> ask first) a file with the current model, which includes all of the figures currently defined except those in the cut buffer. Note that a model file records which atoms are selected and which atom is marked.
- **File -> Save Model As...** is like **Save Model** except that it displays a file browser to save to a different file than the file from which the model was read.
- **File -> Export to Image...** writes a binary "scan-converted" image of the current model to an image file (.raw3) containing a pixel value of 1 in voxels on or internal to the model's implied boundary and 0 elsewhere. This is used to gather statistics of the fit of the model to the grayscale image. The image has the same dimensions and pixel size as the loaded image, or some default if no image is loaded.
- **File -> Export Distance Map** is for testing purposes. Ignore it.
- **File -> Export to BYU Tiles...** writes a tileset file of the implied boundary for transferring the boundary to the boundary displacement program.
- **File -> Load Image...** replaces the currently loaded image with one from disk. Images can have the filename extensions **.raw3** or **.mha**/**.hma** (for UNC MetaHeader format). Press the file browser's **All Files** button to see the non-raw3 filenames. Since images are read-only (the program cannot change them), the only info that can be lost by replacing the image is the global scale from the unit cube to cm. The display is reset as well: each of the model view's 3 displayed slice locations are reset to display the center slice in each dimension. [This should not happen, I think -gst.]
- **File -> Load Tile Set...** replaces the currently loaded tile set with one from disk. The tiles are scaled to the same unit box as the image (if any) or to the unit bounding box of the tiles.
- **File -> Exit** - terminates the program. It will ask if you want to save changes to the current model first, if changes were made.

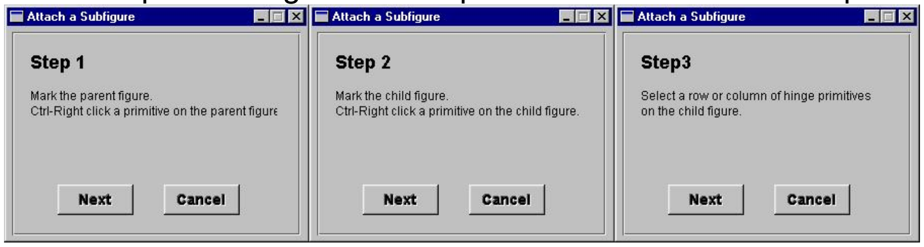
10.2 Main Window -> Edit



- **Edit -> Undo** reverses the last edit made to atoms, as if the last edit never occurred. Almost all edits can be undone, even **Add Quad Figure...**, **Remove Selected Figures...**, **Atom Editor** and **Load Model**. Changing the camera view is not considered an edit and so cannot be reversed - this would have been useful for moving the camera through a scene reproducibly. Multiple edits, up to 1000, can be undone in succession [TECH: and are stored in what is called the ***undo buffer***].
- **Edit -> Redo** reverses the effect of an **undo.** [TECH: the Undo buffer is unchanged by Redo, that is, a Redo does not count as a change to the atoms.]



- **Edit -> Add Quad Figure...** adds figures to the model and assigns colors and arbitrary text names to figures. **Rows** and **Columns** define the **N** and **M** mesh dimensions seen in **Concepts**, page 5. Click **Add** to create a figure – the dialog box stays open so that multiple figures can be added efficiently. The last figure created is left selected so create a figure, move it a little, create the next figure, repeat.
- **Edit -> Attach a Subfigure** displays a sequence of windows to merge 2 independent figures into a parent and child relationship:



Step 1

Mark the parent figure.
Ctrl-Right click a primitive on the parent figure

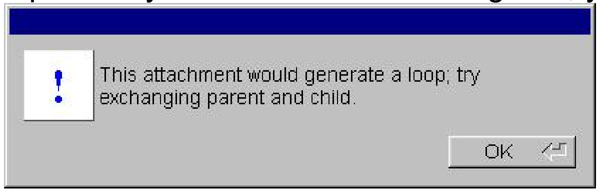
Step 2

Mark the child figure.
Ctrl-Right click a primitive on the child figure.

Step3

Select a row or column of hinge primitives on the child figure.

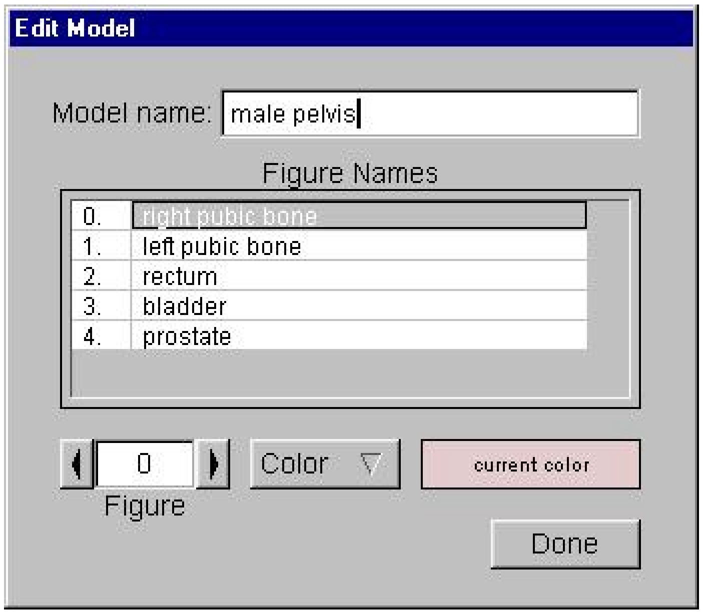
If a child-parent relationship already exists between the 2 figures, you may get this:


This attachment would generate a loop; try exchanging parent and child.

[**Important bug**: the child's hinge atoms may attach to either the parent's EndPrimitive atoms or StandardPrimitive atoms, but not both. In other words, if you think of the parent as a slab, the child may attach to one and ***only*** one of: the slab top, the slab bottom or the slab side. This is a design flaw that we plan to fix. Pablo may crash if you disregard this rule, so save your model just before you attempt to attach each subfigure.]

- **Edit -> Remove Selected Figures...** deletes figures. Select all the atoms in a figure, then click this and poof! the figures are gone. See **Edit -> Undo**.

- **Edit -> Model Properties** displays a dialog box to edit the names of the model and figures and choose a color for each figure. Figures may also be named and colored as they are created with **Add Quad Figure**. This dialog box blocks the interface (no other windows will respond) until **Done** is pressed.



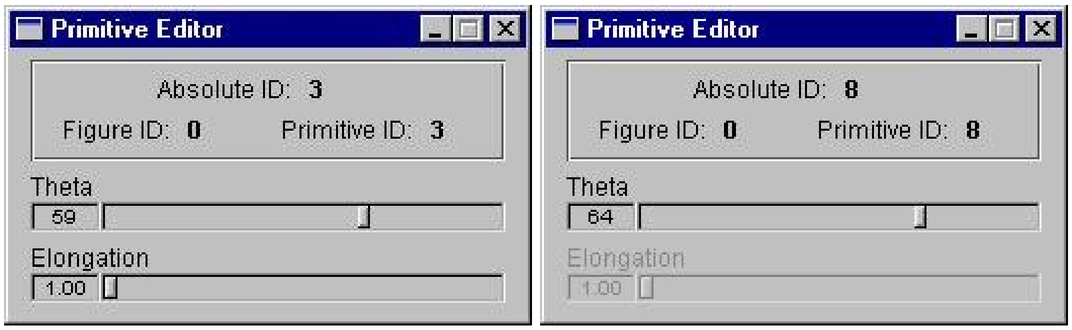
Model name: male pelvis

0.	right pubic bone
1.	left pubic bone
2.	rectum
3.	bladder
4.	prostate

Figure: 0

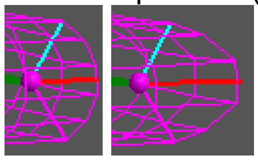
- **Edit -> Select All** selects all atoms in all figures regardless of their current selection state.
- **Edit -> Deselect All** like Select, but deselects instead.
- **Edit -> Toggle All** selects all unselected atoms and deselects all selected atoms. This is useful to perform some operation on a set of atoms and then another operation on the remaining atoms.
- **Edit -> Selection Type** controls what can be selected by the control-left mouse button in the model view window. To select an entire figure with each click, set the selection type to **Figure** and select one or more atoms with the mouse and you will see that the entire figure is selected (or deselected). You can also select/deselect multiple figures easily and quickly using the mouse's marquee feature to affect all figures containing atoms within a 2-D rectangle from the current camera angle. To select/deselect by atom, set the selection type to **Atom**.

- **Edit -> Atom Editor** displays the atom editor:



The Atom Editor is the only place to manipulate **Theta** and **Elongation**, as there is no mouse-oriented manipulator in the model view window. You use these sliders by marking an atom and watching the 3D window, for feedback, as you move the sliders. This window updates as you change which atom is marked. Only EndPrimitive atoms have an elongation (left image, above), so StandardPrimitive atoms have a dimmed elongation slider (right image, above).
The Atom Editor also reports the **Figure ID** and **Atom ID** of the marked atom and it's figure, which are useful when editing model <u>files</u> by hand, see **Editing Model Files**, page 26. **Absolute ID** is a single number that identifies an atom across all figures, eg, if figure 0 contains atoms 0-8, figure 1 would start it's absolute atom IDs from 9. Users may need to edit model files by hand because not all of the functionality required to assign attributes (eg, polarity) are implemented in the user interface.

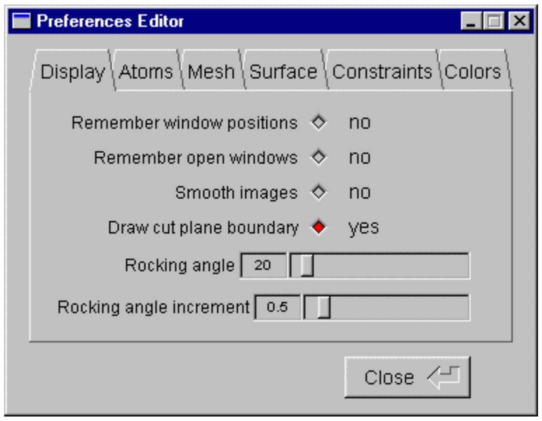
A theta of 90 degrees spreads the involute arms as far as they will go. 0 means the arms overlap. Elongation is 1 (left image, below) for a perfectly round EndPrimitive atom and increases as the edge of the figure gets more pointed (right image):



- **Edit -> Copy** places a exact duplicate of the selected figures into the cut buffer (a store of hidden figures). See **Paste**, below.
- **Edit -> Paste** duplicates the figures in the cut buffer as selected figures. As opposed to most other programs, Pablo Paste does ***not*** replace currently selected figures but instead de-selects them and selects the newly pasted figure(s). Copy and Paste can be used to accumulate figures across m-rep models (in place of editing the files containing those models) or to split a model's figures into separate model files. [Possible bug: I have not tested if subfigure attachments and object constraints are correctly deleted when removing a figure containing these. -gst]
- **Edit -> Mirror** reflects the selected atoms around the center X coordinate (X = .5), but does not affect the Y & Z coordinates. In other words, the atoms are moved to a new position based on each one's current X position only. This is useful because the bilateral symmetry of vertebrates can be used to model a structure on one side of the body and mirror it as a starting place to model the partner structure on the other side of the body. To reflect around a different axis, we suggest: rotate the figures so that the plane of reflection is at X = .5, mirror the atoms, then rotate back to the original orientation.

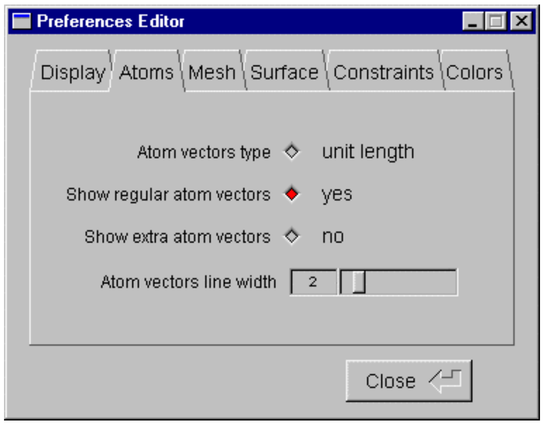
- **Edit -> Constraints** is to set or inspect object to object geometric constraints. [LATER: expand this when new interface is ready!!!]

- **Edit -> Preferences** edit window positions, display attributes, etc. This information is stored in a file in your home directory. The filename is printed when Pablo starts up.



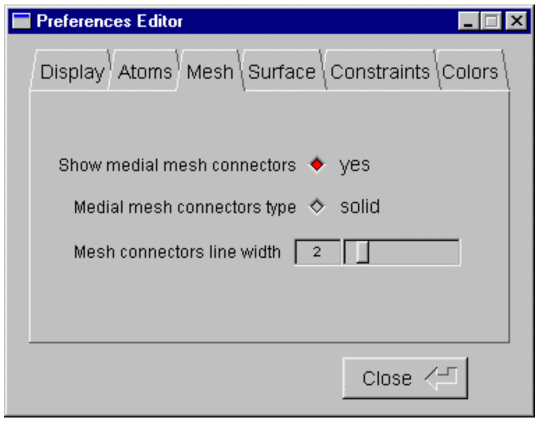
Remember window positions ◇ no
Remember open windows ◇ no
Smooth images ◇ no
Draw cut plane boundary ◆ yes
Rocking angle 20
Rocking angle increment 0.5

Remember window positions records the current size and position of each Pablo window for the next time you start Pablo.
Remember open windows notes which windows are displayed and which are hidden for the next time you start Pablo.
Smooth images changes the model view window's grayscale image display from nearest neighbor interpolation to tri-linear interpolation, making the images look smoother.
Draw cut plane boundary enables that display of a white line at the edges of the X, Y, and Z grayscale cut planes in the model view window
Rocking Angle is the number of degrees that the camera will rock (rotate from side to side) whenever the mouse stops moving.
Rocking angle increment is the rotational degrees to move the camera for each frame of the rocking sequence. Larger numbers appear to rotate the camera faster for the same number of frames.



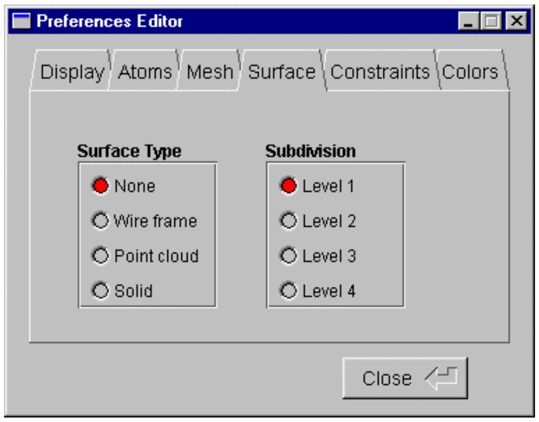
Atom vectors type ◇ unit length
Show regular atom vectors ◆ yes
Show extra atom vectors ◇ no
Atom vectors line width 2

Change the model view window display:
Atom vectors type selects the display of the red **B** vectors from involute length to (elongation * involute length).
Show regular atom vectors draws the B and involute vectors
Show extra atom vectors draws the atom's frame (X, Y, Z vectors)
Atom vectors line width adjusts the atom's vectors thickness. It is useful for illustrations and slide shows.



Show medial mesh connectors ◆ yes
Medial mesh connectors type ◇ solid
Mesh connectors line width 2

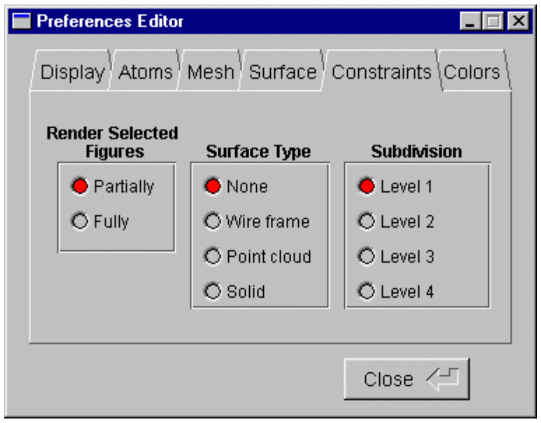
Show medial mesh connectors draws the green lines between atoms
Medial mesh connectors type selects solid, dashed or dotted connectors. It is useful for grayscale illustrations in which color cannot be used to distinguish lines.
Mesh connectors line width adjusts the thickness of connectors.



Surface Type: [✓ None] Wire frame, Point cloud, Solid

Subdivision: [✓ Level 1] Level 2, Level 3, Level 4

Surface Type selects the visualization type of the implied boundary
Subdivision implies the number of times the space between neighboring atoms will be divided to form either the meshed or tiled surface. Higher levels mean finer surfaces that also take more time to compute.

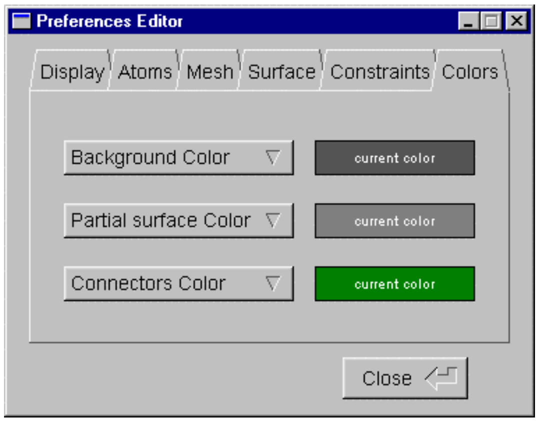


Render Selected Figures: [✓ Partially] Fully

Surface Type: [✓ None] Wire frame, Point cloud, Solid

Subdivision: [✓ Level 1] Level 2, Level 3, Level 4

Render Selected Figures affects the constrained figure's surface visualization during constraint editing (see **Edit -> Constraints**). **Partial** draws the patches of the governed surface which are closer to the governor than the distance slider and does not draw more distant patches. **Fully** uses color changes: the closer portion is drawn in the **Partial Surface Color** (see **Edit->Preferences**) and the farther portion is drawn in the figure's assigned color.
Surface Type is the visualization used for the closer portion.
Subdivision is used for the closer portion.



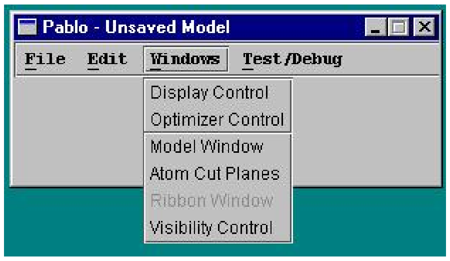
Background Color: current color

Partial surface Color: current color

Connectors Color: current color

Background Color is the background of the model view window. For illustrations, some model colors work better with light background colors, other with dark background colors.
Partial surface Color See the **Constraints** tab, just above.
Connectors Colors is the color is the lines drawn between atoms in the model view window.

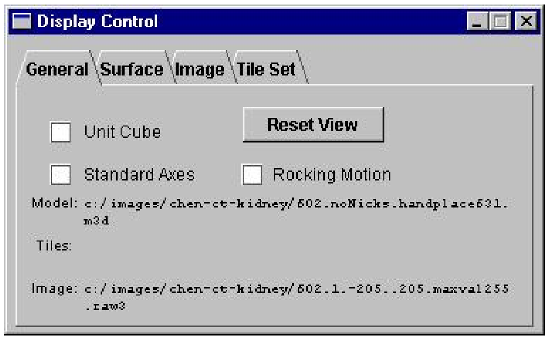
10.3 Main Window -> Windows



The **Windows** menu shows or hides other windows in the application, which are described in this section.

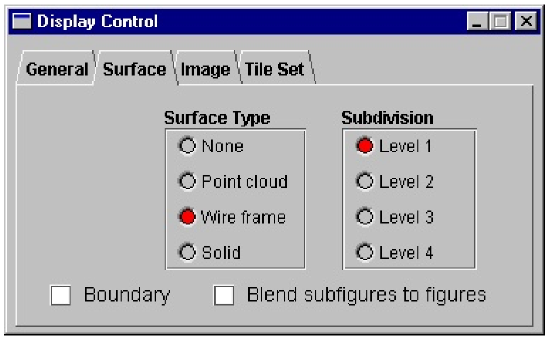
10.3.1 Windows -> Display Control

A tabbed notebook window for controlling the appearance of objects in the 3D model window:



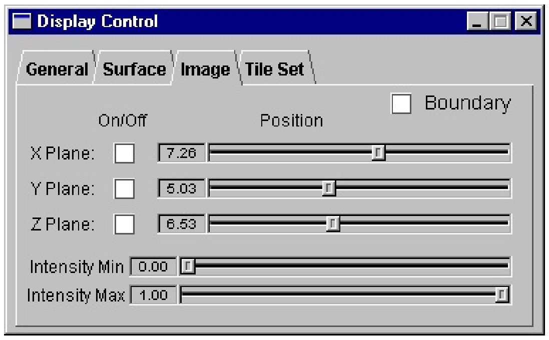
Unit Cube [ ]
Standard Axes [ ]
Rocking Motion [ ]
Model: c:/images/chen-ct-kidney/602.noNicks.handplace631.m3d
Tiles:
Image: c:/images/chen-ct-kidney/602.1.-205..205.maxval255.raw3

Unit Cube shows white edges of the space.
Standard Axes shows the primary X, Y, Z colored axes.
Rocking Motion continually rotates the camera. Good for capturing smooth model animations.
Model, **Tiles** and **Image** show the most recently loaded files. **Undo** does not affect these.
Reset View sets the pose/position/scale of the camera to their initial values.



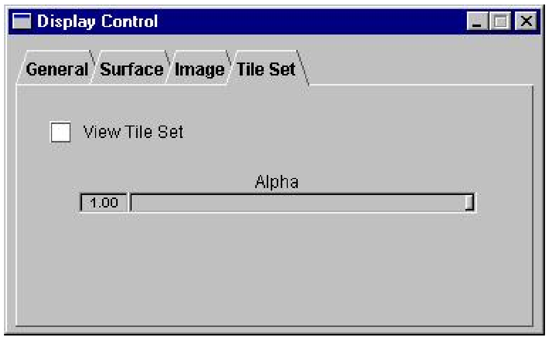
Surface Type: None / Point cloud / [✓ Wire frame] / Solid
Subdivision: [✓ Level 1] / Level 2 / Level 3 / Level 4
Boundary [ ]
Blend subfigures to figures [ ]

Surface Type displays a surface interpolated from the atoms as dots, wire frame, or a tiled opaque surface. **None** hides the surface.
Subdivision is the density of the dots or tile vertices. Higher levels yield a smoother surface.
Boundary shows the 2D intersection of each displayed image X-Y-Z plane with the model's surface.
Blend subfigures to figures shows the real surfaces at hinges where they pull away from either figure.



Boundary [ ]
X Plane: [ ] 7.26
Y Plane: [ ] 5.03
Z Plane: [ ] 6.53
Intensity Min 0.00
Intensity Max 1.00

The **Image** tab controls image display and, surprisingly, the image match term computed in optimization.
Boundary (same button as in **Surface** tab)
On/Off shows any of the 3 image planes.
Position of each plane, in patient coordinate space, selects which 2D slice through the 3D image to draw.
Intensity Min/Max performs clamped intensity windowing: pixel values below **Min** are drawn black, values above **Max** are drawn white, and values between the two are linearly mapped to a grayscale ramp. Min larger than Max is undefined. These are in unit color space and so depend on the pixel values actually present in the image and so might not be portable across images. The image match term is computed from the post intensity windowed images, not from the original image intensities, to allow manipulation of the target image on the fly.



View Tile Set [ ]
Alpha 1.00

View Tile Set shows the tileset surface
Alpha is the tileset's transparency. Tiles do not interact well with image planes in that the tiles may disappear if an image plane is in front of it.

10.3.2 Windows -> Optimizer Control

Fits a model (with an optional training image) to a target image. The user-guided sequence of optimization "stages" (methods) is presented to you in decreasing scale until the atom deformation stage. Boundary displacement is separate and not discussed here. The sequence varies based on what's in the current model, described in detail below:

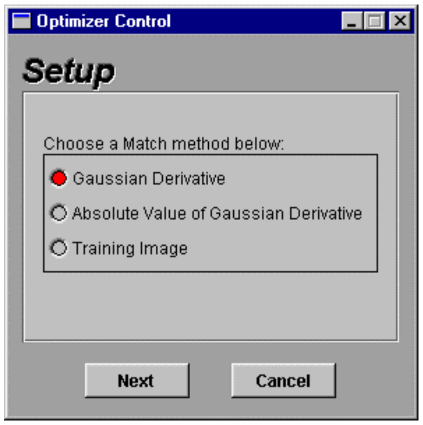
1. Setup - all models (required)
2. Model Stage - all models (optional)
3. If model has multiple figures:
    a. Main Figure Stage (optional)
    b. Atom Stage - for main figure only (optional)
4. Repeat, once per sub-figure:
    a. SubFigure Stage (optional)
    b. Atom Stage - for sub-figure only (optional)

Each figure gets evaluated down to the figure's atom stage before starting on the next figure, that is, the second figure gets it's subfigure and atom stages before the third figure gets it's subfigure and atom stages.

Atom selection is managed by Pablo to indicate to the user which figure is being optimized. ***Only the selected atoms may move, so select them all before starting optimization.*** Since atom selection will be managed by the program, the user does not have to know the order of objects, but the user may control which atoms are to be optimized in some stages by selecting them before the optimization setup. The user also has control over which stage to perform next - that is, to repeat the previous stage, cancel out of all stages and go back to the beginning, or proceed to the next stage in the order listed above - so as to guide the program when it does not find the absolute maximum of it's objective function.

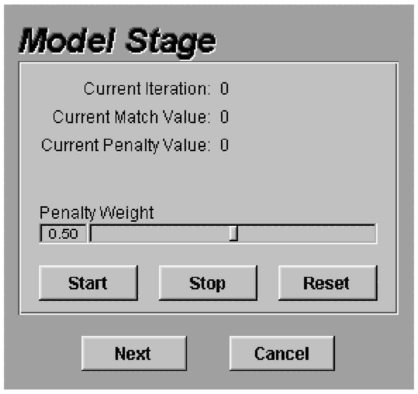
All but the atom stage iterate over changes in a similarity transform until a local maximum is found, seeking the best position, pose, isotropic scale and elongation (along the largest atom dimension) for the model, figures or atoms that matches the image but is penalized for moving too much from the original position/pose/scale both in terms of the image space and of object-object space. Press **Stop** if the model goes awry and Pablo will pause after the current iteration, which could be seconds to minutes.

In all **Optimization Control** dialog boxes, **Next** displays the next pane, chosen in the sequence as explained above, and **Cancel** stops optimization and hides the **optimization control** window. The optimization stages are, using stage numberings from above:

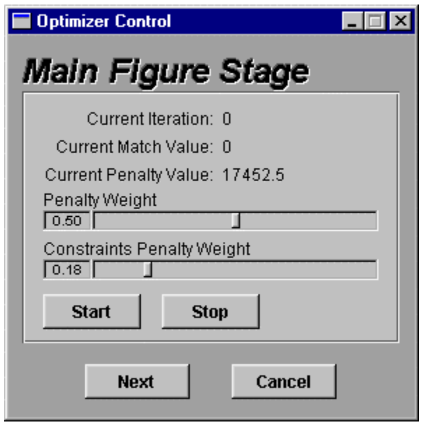

Optimizer Control
Setup
Choose a Match method below:
Gaussian Derivative
Absolute Value of Gaussian Derivative
Training Image
Next
Cancel

1 - Setup specifies the image match to be used for all the optimizations that follow, until **Cancel** is pressed. Intensity windowing settings are used (see **Display Control -> Image** tab), but this seems to introduce noise into our objective functions and so it is recommended that a wide-open window be used (0 .. 1).
Gaussian Derivative is for lighter objects on darker backgrounds, or vice-versa if ***positivePolarity*** is 1.
Absolute Value of Gaussian Derivative is the same as above

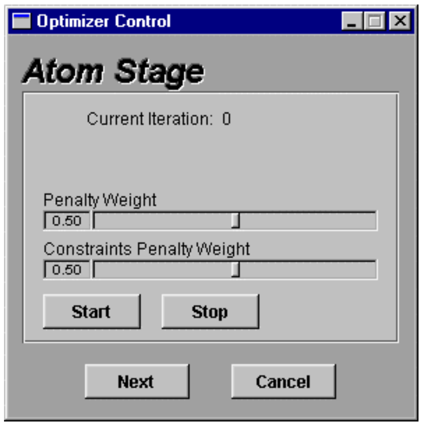
but does not care which side is darker as long as there is a difference in intensity.
Training Image correlates a second image with the currently loaded image in a collar around the implied boundary. Click **Next** and Pablo will prompt for the training image's filename. This filter responds well under almost any condition but is slower than the other methods.



2 - Model Stage fits the model to the image using a similarity transform (pose, position, isotropic scale) plus elongation (along the longest figure axis as measured by the highest atom count).
Penalty Weight is the exponential strength of the geometric penalty that pegs the model close to it's starting position/pose/scale. Decrease the weight to allow the model to wander more.
Start begins iterating the fitting, using the current position/pose/scale as an anchor from which the model may not wander too much. Iterations will continue to accumulate until no more improvements are seen.
Stop temporarily ceases iterating the fitting at the next possible time. Resume from where it stopped by pressing **Start** once again.
Reset uses the current position/pose/scale as the starting position/pose/scale and may take up to 10 seconds to complete. Uses: you move the model, eg, because it wandered too much so you pressed **Stop**; you want to perform a second registration using a prior registration as a starting position/pose/scale, eg, you want to change the Penalty Weight halfway through the registration.



3.a - Main Figure Stage is the same as **Model Stage** except that just figure 0 may move and object-object constraints are added. The **Constraints Penalty Weight** .slider controls an exponential weighting factor applied. Higher numbers give more influence to the object-object constraints.



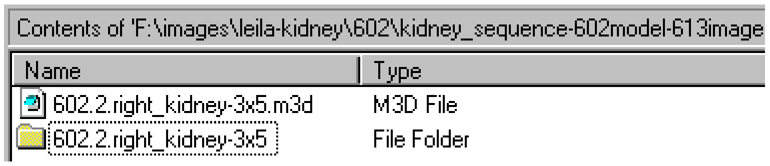
3.b – Atom Stage allows each atom in figure 0 to vary in position, pose, r (length of B vector) and theta (angle between B and Y vectors). Atom elongation is not currently optimized but is used to position of the implied boundary surface.

Stages for sub-figures are the same as for the main figure but add a slider for constraints penalty weight that balances it against the similarity transforms penalty weight.

Auto-save - to review the model as it changes during fitting.

The features in the blue section below must be enabled by a programmer

The model is written after each iteration. Enable before pressing **Start** by making a (folder) in the model's folder that's named the same as the model except for the ".m3d" extension:



Contents of 'F:\images\leila-kidney\602\kidney_sequence-602model-613image

Name	Type
602.2.right_kidney-3x5.m3d	M3D File
602.2.right_kidney-3x5	File Folder

In this lower folder will be written a series of model files, one after each iteration. Files are named **registration_0001.m3d**, etc. The filename's number is the iteration. *Pablo* will not write models if the directory does not exist and it will overwrite old model files.

10.3.3 Windows -> Model Window

Displays the 3D model window. See Mouse Buttons for more interactions.

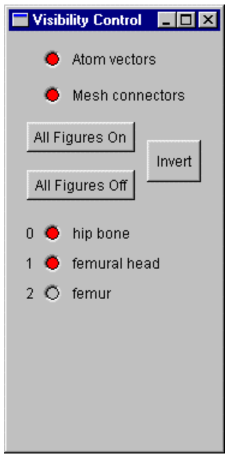
10.3.4 Windows -> Atom Cut Planes

Displays checkboxes for the 6 atom Cut Plane windows. See Slice View Windows. These windows may be drawn slowly and they do not respond to the mouse.

10.3.5 Windows -> Ribbon Window

[*DIMMED*: displays the Ribbon window. It is not clear if this window is useful yet.]

10.3.6 Windows -> Visibility Control



Sets visibility of all atoms' vectors and mesh connectors, and individual figures. Use to simplify the model view: to temporarily remove obscuring figures, to limit atom selection mouse actions to specific figures, for illustrative purposes, to teach model building. Figures exist even when hidden. [BUG: the marked atom is visible when in hidden figures.]

Atom vectors draw the B and involute vectors on each atom.
Mesh connectors draw the lines between atoms
All Figures On/Off turn all of the figures attributes visible or invisible.
Invert: visible figures are hidden and invisible figures are drawn.

The bottom section indicates each figure's visibility. Each line has the figure's ID, visibility, and each figure's name, if it exists. Names are assigned by **Edit -> Add Quad Figure...**, section 10.2.

11 Console window

Text messages are printed here mostly as diagnostics. Users can view the filenames loaded and saved and the file Pablo is seeking for your **preferences**.

12 Editing Model Files

Model files need to be edited by hand with a text editor (vi, emacs, Notepad, Word, etc) because some features are not yet supported within the user interface. Model files use a nested folder format, similiar in concept to files on disks, called the ***PaulY nested folder*** format. which is explained best with a sample in hand:

```
model {
   figureCount = 5;
   figureTrees {
      count = 2;
      tree[0] {
         blendAmount = 0;
         blendExtent = 0;
         childCount = 1;
         figureId = 0;
         linkCount = 0;
         child[0] {
            blendAmount = 0.5;
            blendExtent = 0.5;
            childCount = 0;
            figureId = 1;
            linkCount = 2;
            link[0] {
               primitiveId = 4;
               t = -1;







      }
   }
```

…continued in next column…

Continued from last column…

```
   figure[0] {
      name = hip bone;
      numColumns = 5;
      numRows = 3;
      positivePolarity = 1;
      positiveSpace = 1;
      type = QuadFigure;
      color {
         blue = 0.1;
         green = 0.2;
         red = 0.8;
      }
      primitive[0][0] {
         elongation = 1.2;
         qw = 0.889735;
         qx = -0.228895;
         qy = 0.388902;
         qz = 0.0687963;
         r = 0.0276761;
         selected = 1;
         theta = 72.4705;
         type =EndPrimitive;
         x = 0.413206;
         y = 0.339683;
         z = 0.123524;
      }
      primitive[0][1] {
         elongation = 1.2;
         qw = 0.989139;
         ...snipped...
      }
   }
}
```

A **folder**, in curly brackets associates some values together and keeps other apart:

```
name {
  contents...
}
```

In the sample above, `model` is the name of a folder whose contents are delimited by the first `{` and last `}` (color-coded in red above). Note that you must write the `{` on the same line as the folder's name, and that the `}` must be on a line by itself.
Folders may be nested within other folders. The model folder contains a blue folder for

`figureTrees` and a orange folder for each. Each figure folder in turn holds folders for the figure's color and for each of the figure's atoms.

Values - a value is assigned to a name by an assignment line of the form:

name = *value*;

The names are reserved keywords - only a programmer may create new names. The values can be integers or real numbers or even short words, but all assignment lines must end with a semicolor (`;`). That assignment is specific to the folder in which one finds the assignment line, for instance, the `elongation` you assign to `primitive[0][0]` will not be confused with the `elongation` for `primitive[0][1`] because of the context, even though both lines look identical:

elongation = 1.2;

For multiple instances of a name or folder to exist within the same folder, they must be distinguished by one or more trailing numbers, each in square brackets, eg, `figure[0]` or `primitive[0][1].` In Pablo, this number is the figure ID when used in the first instance, and the index to the atom is the figure-specific atom id when the atoms are sorted in alphabetical order. An example of the latter for a 3-row, 5-column model:

primitive[0][0] = ID 0	primitive[1][0] = ID 5	primitive[2][0] = ID 10
primitive[0][1] = ID 1	primitive[1][1] = ID 6	primitive[2][1] = ID 11
primitive[0][2] = ID 2	primitive[1][2] = ID 7	primitive[2][2] = ID 12
primitive[0][3] = ID 3	primitive[1][3] = ID 8	primitive[2][3] = ID 13
primitive[0][4] = ID 4	primitive[1][4] = ID 9	primitive[2][4] = ID 14

Robust & flexible - values and folders may be missing and may be assigned default - but not necessarily useful - values by the program so that it will not crash. This is useful so that cleverly written programs can add keywords to the format and still allow old model files to be read with reasonable results, although this is not always possible. Properly formatted extra values, unrecognized by the program, will be silently ignored, again not crashing the program. This is useful if multiple programs will be updating the model file with their own values, perhaps that only the authoring program would care to understand.
The lines that you may need to edit by hand are listed in the sample model file above. Details:

- color - each figure may have a color assigned to it. Colors are represented as combinations of various amounts of Red, Green and Blue in the unit color space. Some examples:

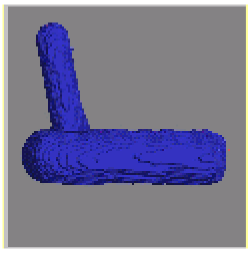
`red=1;`
`green=0;`
`blue=0;`
	The color red

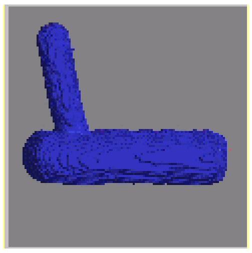
`red=0;`
`green=1;`
`blue=1;`
	Cyan is equal parts green and blue

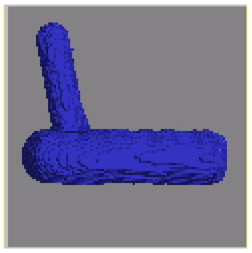
`red=1;`
`green=.5;`
`blue=0;`
	Orange has twice as much red than green

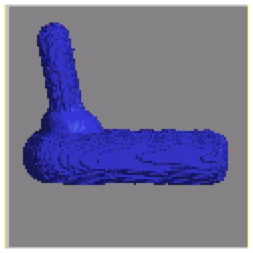
  Unspecified color components default to those of cyan. Assign this inside each figure's folder.

- `figureTrees` - are required to perform any type of fitting (optimization) or to display blending surfaces. See **Attach a Subfigure**, section 10.2**,** for the new user interface to these features. Each tree represents a hinge: a set of connections between pairs of a ***parent*** coordinate (u,v,t) and a ***child*** atom. It is required that the child figures' atoms at the ***hinge*** be positioned somewhere on the parent's surface. The parent and child figure IDs are the ***N***'s from the "figure[***N***]" folder names. The figureTrees folder is in the model folder.
  In the figureTrees folder are tree[***N***] folders, where you'll specify the `blendAmount` and `blendExtent` used for generating a surface both for display and for generating the geometric match (currently we calculate distance from the blended surface as one of the inputs) when blended figures are fitted. blendExtent indicates how far into the figures should be blended. For example, the web between human fingers is a low amount and the web between a duck's fingers is a high amount. blendAmount indicates the amount of bulge in the blend. Real numbers [0..1]. Visually:


blendAmount=0
blendExtent=0


blendAmount=0
blendExtent=1


blendAmount=1
blendExtent=0


blendAmount=1
blendExtent=1

- `polarity` - both Gaussian image matches assume that the image intensities follow a pattern of bright object on a dark background (**`positivePolarity = 1`**;) or dark object on a bright background (**`positivePolarity = 0;`**). The background includes all the objects in the scene (even air) that are not the object. Although this is simplistic, it works in many cases. Assign this inside each figure's folder.
- `positiveSpace` **-** [LATER: 1 for a protrusion figure (figure is a solid part of the object) and 0 for indentations (figure is a hole in the object).]

13 Items to document

- Section 9.2: **Slice View windows** needs a picture
- Section : **Edit -> Constraints** should be

14 Index

Bold entries indicate buttons or window titles. Other entries are concepts or actions.