

MSMS User's Manual

**Medical Device Development Facility
Biomedical Engineering Department
University of Southern California**

**By:
Rahman Davoodi**

Version 0.9.2 Beta

Revision Sheet

| Release No. | Date | Revision Description |
|-------------|----------|--|
| Rev. 0 | 09/05/06 | First draft |
| Rev. 0.5 | 12/08/06 | <ul style="list-style-type: none"> The model is now subdivided and categorized into human, prosthesis and world models each defined in separate XML files. World objects are treated as segments connected to the world via 6 DOF joints. A primitive skinning tool covers the discontinuity in joints of the human limb by a patch. The model can now be closed to load a new model. Segments now have collision properties. Multiple images of mesh or primitive type can now be assigned to each segment. Each image can be independently edited. |
| Rev 1.0 | 03/14/07 | XML description of segment and joint components. |
| Rev 1.1 | 05/07/07 | MSMS 0.6.0 <ul style="list-style-type: none"> The model cannot be closed. |
| Rev 1.2 | 05/30/07 | MSMS 0.6.1 <ul style="list-style-type: none"> New Open dialog. Models can be closed. |
| Rev 1.3 | 06/07/07 | MSMS 0.6.1 <ul style="list-style-type: none"> Added descriptions on properties for all component types. |
| Rev 1.4 | 08/17/07 | MSMS 0.7 <ul style="list-style-type: none"> Description of new features since 0.6.1 |
| Rev 1.5 | 11/6/07 | MSMS 0.7.3 Beta <ul style="list-style-type: none"> Description of new features Rules for importable SIMM and SolidWorks models |
| Rev 1.6 | 11/15/07 | MSMS 0.7.5 Beta |
| Rev 1.7 | 07/01/08 | MSMS 0.7.5 Beta <ul style="list-style-type: none"> Description of how to use the streaming data C S-function. Section 11. Description of how to modify the Simulink model to animation using 'Ordered Joint Angles'. Section 11. |
| Rev 1.8 | 03/06/08 | MSMS 0.7.8 Beta <ul style="list-style-type: none"> Description of save images feature. Description of add model feature. Description of color and light in MSMS (Section 12). |
| Rev 1.9 | 04/18/08 | MSMS 0.7.8 Beta <ul style="list-style-type: none"> Muscle Tendon Length. |
| Rev 2.0 | 03/03/09 | MSMS 0.8.6 Beta <ul style="list-style-type: none"> Description of ADL animation Description of new motion file format (msm) |
| Rev 2.1 | 04/21/09 | MSMS 0.8.7 Beta <ul style="list-style-type: none"> Fix to ADL animations for EMG-Animation synchronization Modifications to motion file to explicitly represent pause slides |

| | | |
|---------|----------|--|
| | | <ul style="list-style-type: none">• Fix to allow selection/unselection of objects by mouse clicks• New icons in the toolbar for camera controls• Modifications to allow installation of MSMS in Linux |
| Rev 2.2 | 05/19/09 | MSMS 0.8.8 Beta <ul style="list-style-type: none">• Complete review of the user manual for compatibility with MSMS and addition of new appendices• A new section on Feature Commands animation• A new section on adding images to the segments |
| Rev 2.3 | 07/14/09 | MSMS 0.9.0 Beta <ul style="list-style-type: none">• New Model Info menu item• A new appendix on special features |
| Rev 2.4 | 10/28/09 | MSMS 0.9.1 Beta <ul style="list-style-type: none">• Fixed issues with head tracking feature command |
| Rev 2.5 | 6/25/10 | MSMS 0.9.2 Beta <ul style="list-style-type: none">• Upgraded to Java 6 update 20• Upgraded to Java3D 1.5.2• Fixed issues with Windows 7 installation |

Table of Contents

| | |
|---|----|
| 1. General Information..... | 7 |
| 1.1. Introduction..... | 7 |
| 1.2. Minimum System Requirements | 7 |
| 1.3. Installation Procedure | 7 |
| 1.4. Key Features | 7 |
| 2. Starting MSMS | 8 |
| 2.1. Using Command Line Interface | 8 |
| 2.2. Using Windows Shortcuts | 8 |
| 2.3. Overview of MSMS GUI | 8 |
| 3. File Menu | 10 |
| 3.1. New Model | 10 |
| 3.2. Open | 10 |
| 3.3. Reopen | 11 |
| 3.4. Save..... | 11 |
| 3.5. Save As | 11 |
| 3.6. Save Simulation | 11 |
| 3.7. Close | 12 |
| 3.8. Import Model | 12 |
| 3.8.1. Import SolidWorks Model | 12 |
| 3.8.2. Import SIMM Model | 14 |
| 3.9. Exit | 15 |
| 4. View Menu..... | 16 |
| 4.1. Points of View | 16 |
| 4.1.1. Front | 16 |
| 4.1.2. Back | 16 |
| 4.1.3. Right | 16 |
| 4.1.4. Left | 16 |
| 4.1.5. Top | 17 |
| 4.1.6. Bottom (Under)..... | 17 |
| 4.2. Cameras | 17 |
| 4.3. Camera Light On | 18 |
| 4.4. Default Lights On | 18 |
| 4.5. View All..... | 21 |
| 4.6. Full Screen Mode..... | 21 |
| 4.7. On-Screen Text Display..... | 21 |
| 4.8. Display Setup | 21 |
| 4.8.1. Display | 22 |
| 4.8.2. Framing | 23 |
| 4.8.3. User | 23 |
| 4.8.4. Head Tracking..... | 24 |
| 5. Model Menu | 25 |
| 5.1. Move (translate) | 25 |
| 5.2. Show All Axes | 25 |
| 5.3. Show Ground Axes | 25 |
| 5.4. Show Joint Sliders | 25 |
| 5.5. Add Component | 25 |
| 5.5.1. Segment Appearance..... | 26 |
| 5.6. Add Model | 27 |
| 5.7. Remove Component | 27 |
| 5.8. Scaling..... | 27 |

| | |
|---|----|
| 5.9. Model Info | 27 |
| 6. Simulation Menu | 29 |
| 6.1. Simulation setup | 29 |
| 6.1.1. General | 30 |
| 6.1.2. Setup | 30 |
| 6.1.3. Solver | 30 |
| 6.1.4. Dynamic Engine | 31 |
| 6.1.5. Output Data | 31 |
| 7. Animation Menu | 32 |
| 7.1. Start | 32 |
| 7.2. Pause | 32 |
| 7.3. Stop | 32 |
| 7.4. Rendering Stats | 33 |
| 7.5. ADL File Parsing | 33 |
| 7.6. Write Motion File | 33 |
| 7.7. Setup | 33 |
| 8. Help Menu | 36 |
| 8.1. MSMS User Manual | 36 |
| 8.2. Tutorials | 36 |
| 8.3. MSMS Viewpoint Shortcuts | 36 |
| 8.4. Report Bug | 36 |
| 8.5. Request Feature | 36 |
| 8.6. MSMS on the Web | 36 |
| 8.7. MSMS Discussion Group | 37 |
| 8.8. About MSMS | 37 |
| 9. Appendix A: MSMS Files and Directory Structure | 38 |
| 9.1. Model Library folder | 38 |
| 9.2. Simulations folder | 38 |
| 9.3. Workspace.xml | 38 |
| 9.4. View.xml | 39 |
| 10. Appendix B: Component GUIs | 42 |
| 10.1. Segment | 42 |
| 10.1.1. General | 42 |
| 10.1.2. Inertia | 42 |
| 10.1.3. Material | 43 |
| 10.1.4. Image | 45 |
| 10.2. Joint | 48 |
| 10.2.1. General | 48 |
| 10.2.2. Segments | 49 |
| 10.2.3. Axes | 50 |
| 10.2.4. Motion | 51 |
| 10.2.5. Image | 53 |
| 10.3. Muscle | 54 |
| 10.3.1. General | 54 |
| 10.3.2. Attachments | 54 |
| 10.3.3. Wrap Objects | 55 |
| 10.3.4. Lengths | 56 |
| 10.3.5. Fascicle | 57 |
| 10.3.6. Image | 61 |
| 10.3.7. Dynamics | 62 |
| 10.4. Wrapping Object | 63 |
| 10.5. Kinematic Driver | 63 |
| 10.5.1. General | 63 |

| | |
|---|----|
| 10.5.2. Attachments | 64 |
| 10.5.3. Image | 64 |
| 10.6. Position Sensor | 65 |
| 10.6.1. General | 65 |
| 10.6.2. Attachment..... | 65 |
| 10.6.3. Image | 65 |
| 10.7. Light | 66 |
| 10.7.1. General | 67 |
| 10.7.2. Position & Direction | 68 |
| 10.7.3. Color..... | 69 |
| 10.7.4. Miscellaneous..... | 70 |
| 10.8. Camera | 71 |
| 10.8.1. General | 71 |
| 10.8.2. Position & Orientation | 72 |
| 10.8.3. Frustum | 73 |
| 11. Appendix C: XML Definitions of Model Components | 74 |
| 11.1. Segment | 74 |
| 11.2. Joint | 75 |
| 11.2.1. Comments:..... | 77 |
| 11.3. Muscle | 77 |
| 11.4. Wrapping Object..... | 79 |
| 11.5. Kinematic Driver..... | 79 |
| 11.6. Position Sensor | 79 |
| 11.7. Light | 80 |
| 11.8. Camera | 82 |
| 12. Appendix D: Motion File Formats | 84 |
| 12.1. MSMS Motion File Format (.msm) | 84 |
| 12.1.1. Format of the header row | 84 |
| 12.1.2. Format of the data rows | 85 |
| 12.1.3. An Example msm Motion File | 85 |
| 12.2. SIMM Motion File Format (.mot) | 86 |
| 12.3. Matlab Motion File Format (.mat)..... | 86 |
| 13. Appendix E: Protocol for Live Animation Data | 87 |
| 13.1. Ordered Joint Angles..... | 87 |
| 13.2. Feature Commands..... | 87 |
| 13.2.1. Packet Protocol for Feature Commands | 88 |
| 13.2.2. Example UDP Packets for Feature Commands | 90 |
| 14. Appendix F: ADL Animations | 92 |
| 14.1. Using MSMS and PowerPoint to Animate ADL movements | 92 |
| 14.2. MSMS model | 92 |
| 14.3. Library of the motion files | 92 |
| 14.4. ADL Animation Sequences | 92 |
| 14.5. Rules and Guidelines for Creation of ADL sequence in PowerPoint | 95 |
| 15. Appendix G: Special Features | 96 |
| 15.1. Blanking the model screen..... | 96 |
| 15.2. Writing animation events to file..... | 96 |
| 15.3. Ordered Joint Angles – APL..... | 97 |

1. General Information

1.1. Introduction

MSMS is a software application for modeling and simulation of neural prostheses systems. It can be used to model and simulate human and prosthetic limbs and the task environment they operate in. The simulations can be executed in a standalone computer to develop and test neural control systems or in a virtual reality environment where the human or animal subject can interact with and therefore affect the behavior of the simulated limb.

1.2. Minimum System Requirements

- Operating system: Windows XP.
- 512MB of RAM or higher recommended.
- Processor speed of 1.2GHz or higher recommended.

1.3. Installation Procedure

MSMS is distributed through a single compressed zip file. To install MSMS, unzip the compressed file to get the executable installer. Then run the installer and follow the instructions.

1.4. Key Features

- Standard XML format for model description
- Tools for creation, editing, and visualization of 3D models
- Ability to import existing models from SIMM and SolidWorks
- Ability to model human limbs and prosthetic limbs and the task environments
- Ability to animate models with the data stored in motion files or streamed in real-time from a simulation or motion capture system
- Ability to playback animation sequences created in PowerPoint
- Tools for physics-based simulation of the limb movement and its interactions with the environment
- Tools for creation of complex virtual environments for subject in the loop simulations of the neural prostheses systems

2. Starting MSMS

2.1. Using Command Line Interface

The MSMS command-line syntax is:

```
runmsms [ "Dir_name" [ A [ Live_Data [ Stereo_3D [ Fullscreen ] ] ] ] ]
```

Where,

"Dir_name" = Directory name where the MSMS model resides

NOTE: The directory name *must* be in quotes.

A = Tells MSMS to start in animation mode

Live_Data = Controls head-tracking mode (use arguments *ON* or *OFF*)

Stereo_3D = Controls stereo mode (use arguments *ON* or *OFF*)

Fullscreen = Controls full-screen mode (use arguments *ON* or *OFF*)

Example:

```
runmsms "c:\msms\simulations\Sample Patient\SCN-02-01" a on off off
```

2.2. Using Windows Shortcuts

By default, MSMS places an icon on the desktop to launch MSMS. This will bring up the MSMS application window where the models will be loaded, viewed and edited.

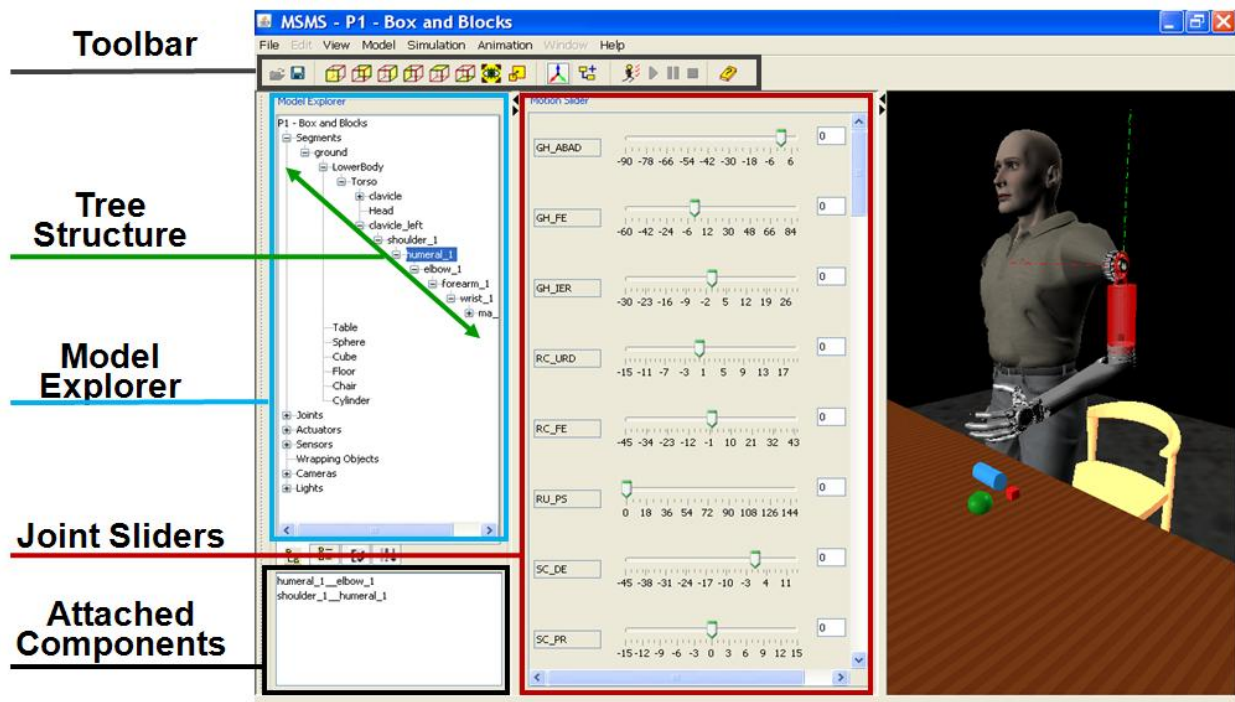
2.3. Overview of MSMS GUI

The MSMS graphic user interface consists of the following sections:

- **Menu Bar** located at the top of the MSMS window, and gives access to most MSMS features.
- **Toolbar** is located just below the Menu Bar and gives access to the most frequently used features such as opening and saving models.
- **Model Explorer** lists all of the components in the opened model. The list can be viewed in four different ways by clicking on one of the four buttons on the Model Explorer. Once a component is selected in the top Pane, its attached components are

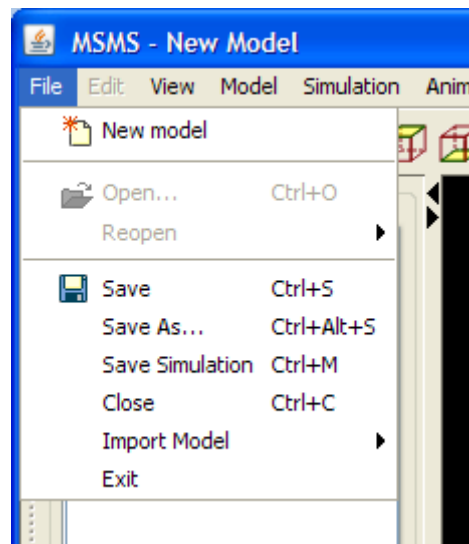
displayed in the bottom Pane. Double clicking on a component's name will bring up its properties dialog box.

- **Joint Sliders** lists all of the degrees of freedom in the model. The sliders can be used to manually move individual degrees of freedom in the model.
- **3D Model Window** is where the MSMS models are visualized, edited and animated.



3. File Menu

This menu runs operations on the model files that serve as input to MSMS. The File menu may be considered to be the entry point to the system.



3.1. New Model

This command creates an empty model, i.e. a model with no components. Once you have created a new model, you can add components to it graphically.

3.2. Open

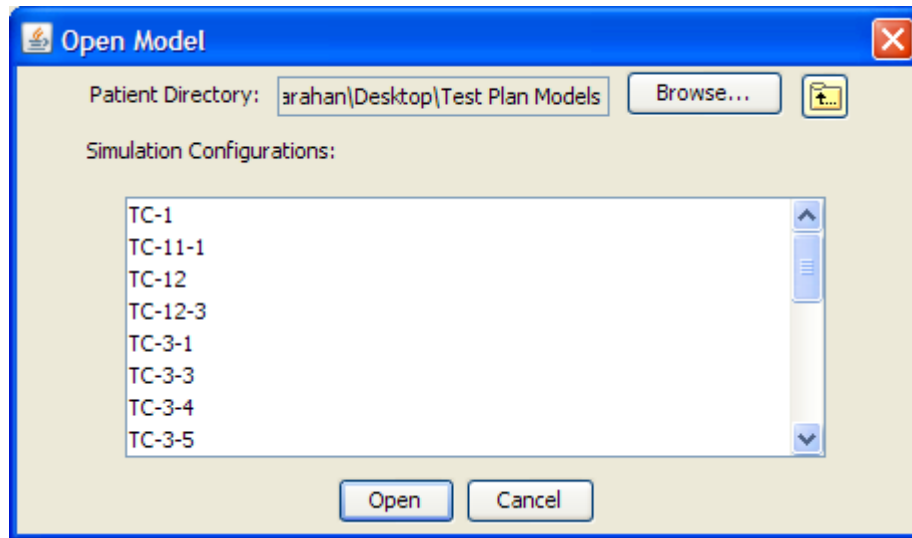
This command loads an MSMS model described in XML format that is saved on disk. A model must, at a minimum, contain the following folders and files.

[Model Folder]\Model\Humans\human.xml
 [Model Folder]\Model\Prostheses\prosthesis.xml
 [Model Folder]\Model\Worlds\world.xml

A model can contain three other folders:

[Model Folder]\Model\Images
 [Model Folder]\Data
 [Model Folder]\Matlab
 [Model Folder]\Setup.

To open a model, click on File > Open > Browse and choose the folder that contains [Model Folder]. This causes the Models in that folder to be listed in the Open dialog. To open a model, click on its name and select Open. To learn more, see Appendix A: MSMS Model Directory Structure.



3.3. Reopen

This command provides a list of five recently opened models and allows the user to quickly open one of them.

3.4. Save

This command saves any user made changes to the model's XML file on disk.

3.5. Save As

This command saves changes to the model under a different name or the same name at another location on disk. If the model contains images that are not stored in the MSMS Image Library folder, these images are copied to the new model location. Furthermore, if the "Save Library Images" option is checked, even the images located in the Image Library are copied to the new model location.

3.6. Save Simulation

This command saves a Simulink simulation model (.mdl) in the MATLAB folder. This simulation model represents the algorithms that can simulate the movement of the MSMS model in response to control excitations and external forces. You can open and run this model in MATLAB's Simulink program. Once run, the simulation model will generate motion data that are saved in a Matlab mat file for later analysis and sent to MSMS via UDP for on-line animation. To view the simulated motion in MSMS while the simulation is running in Simulink, you need to setup and run the animation from the Animation menu.

Note that if any changes have been made to the model since it was opened, saving simulation will also save the model and erase the dat in the Data folder. Therefore, if you plan to save simulation, but don't want the original model to be altered, make a copy of the model first. Then, open the copy, make any changes you desire, and save simulation.

3.7. Close

This command closes the opened model.

3.8. Import Model

3.8.1. Import SolidWorks Model

This command imports a SolidWorks model into MSMS. MSMS allows importing CAD designs from SolidWorks. This process is as follows.

Engineers can build accurate models of prosthetic limbs in SolidWorks and automatically convert it to a Physical Modeling XML file. This conversion is done in the SolidWorks environment using the CAD-to-SimMechanics translator, a free add-on utility available from Mathworks website. The Physical Modeling XML file can be read into SimMechanics to dynamically simulate the prosthetic limb. The same file can also be read into MSMS to represent the prosthetic limb in MSMS. No MATLAB component is required at any time to perform the import to MSMS.

The Physical Modeling XML includes bodies to represent the assembly's parts and maps the constraints between the parts into joints. The 'Part', the 'Constraints' (or mates), the 'Fundamental Root' and the 'Subassembly' Solidworks components correspond respectively to the 'Body', the 'Joints', the 'Ground-Root Weld-Root Body' and the 'Subsystem' in SimMechanics. After importing the prosthetic limb model into MSMS, it can be populated with components such as actuators and sensors, it can be attached to a human model in MSMS, and it can be simulated within an appropriate task environment.

In its import process, MSMS also uses STL files exported from SolidWorks to properly visualize the prosthetic limb segments. Therefore, MSMS integrates a complete set of information directly available from the SolidWorks CAD design software: the assembly and its visualization. The process should be completely automatic and the user should not have to modify the linkage manually in the XML file. This aspect of the translation guarantees the data has not been corrupted.

Because conversion from CAD to SimMechanics involves interpretations of what each element in the CAD model is and how it has to be represented in SimMechanics, Mathworks provides guidance and instructions on configuration and export of your CAD model so that it can be successfully imported into SimMechanics. You can find these guidelines in MathWorks web site at:

<http://www.mathworks.com/access/helpdesk/help/toolbox/physmod/mech/>

Further, current MSMS tools for importing CAD models do not handle all system topologies, which imposes additional constraints on importable CAD models. For example:

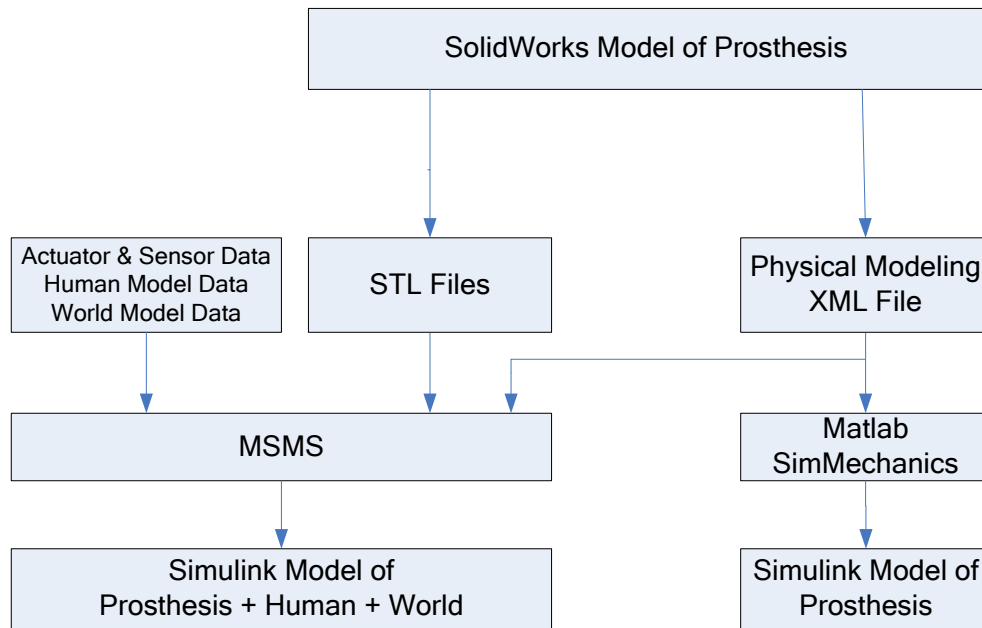
- The MSMS converter does not yet handle closed loop mechanical systems and subassemblies. If the SolidWorks model includes a closed loop system or a subassembly, it cannot get converted to MSMS. Subassemblies can be avoided in the model by bringing the parts and the constraints contained in the subassembly to the higher

hierarchical level. In other words, a flat assembly is required: all parts are mated together at the top level.

- The CAD assembly parts need to have masses and inertia tensors. This may be automatically computed from density and geometry as long as all parts have material properties defined.
- Unnecessary constraints must be avoided because they are translated into many weld joints that complicate the simulation model but are not necessary.
- Only the complete assembly can be exported to XML and not a part of it.

Finally, the user must export the segment images from the CAD model. MSMS uses these images to visualize the imported CAD model. The following rules will ensure that the images are exported properly.

- A 'Coarse' resolution is sufficient for visualization purposes. STL files have very high resolution because they are meant to be used for manufacturing (stereolithography). MSMS reads directly the STL format. However, this format describes only surface geometry of 3D objects and does not include any information about color or texture. It can be useful to convert the STL files to the OBJ format and then include color and texture properties.
- Set the 'Output' parameter to binary. ASCII STL files can quickly become very large. Therefore, the binary STL is a better option.
- Set the 'Units' to meters. The MSMS environment uses metric units and the STL format does not include any information about the units even though the unit may be specified in the comments in the STL file.
- Check "Do not translate STL output data to positive space". For manufacturing, vertices' coordinates will be converted to positive space. However, in MSMS, it is important to keep the coordinates as they are to preserve the concordance between the images and the Physical Modeling XML file.



Importing a SolidWorks model to MSMS: SolidWorks outputs a Physical Modeling XML file describing the mechanical linkage and STL files representing the appearance of each segment. Both sets of outputs are used by MSMS to create an accurate model of the prosthetic limb in MSMS. The final Simulink model can be built via Matlab/SimMechanics or MSMS. But the use of MSMS allows the users to attach the imported prosthetic limb to a human, add additional prosthetic components such as actuators, and simulate it in an appropriate task environment.

3.8.2. Import SIMM Model

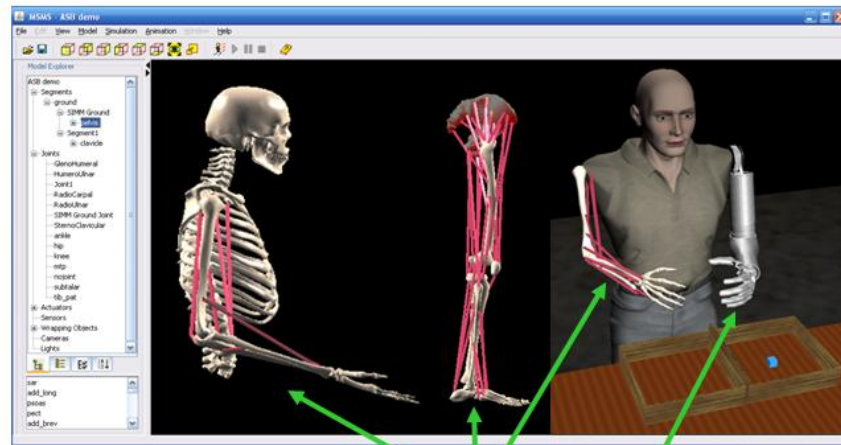
This command imports a SIMM model into MSMS. The Open dialog shows only SIMM model files.

There are some components such as the Torus wrapping objects and muscles via points that are not currently supported by MSMS. These components will be simply ignored by the import process.

Because there are no published specifications for SIMM files, some files may have elements that could cause errors when loaded into MSMS. To successfully import SIMM models into MSMS, follow the following guidelines:

- Every component must have a unique name. Do not reuse component names.
- Define the segments always before defining the joints.
- The “.asc” bone files should be normalized. If you have any bone file that is not normalized (has three numbers for each vertex instead of six), normalize it using the “Norm.exe” utility provided by SIMM.

- If after loading your SIMM model into MSMS the images are displaced, use image manipulation tools in MSMS to properly position and orient the images.
- If you have muscles in your model, make sure that every segment has a bone image. MSMS uses the bone image to constrain the muscle attachment points to bony surfaces.

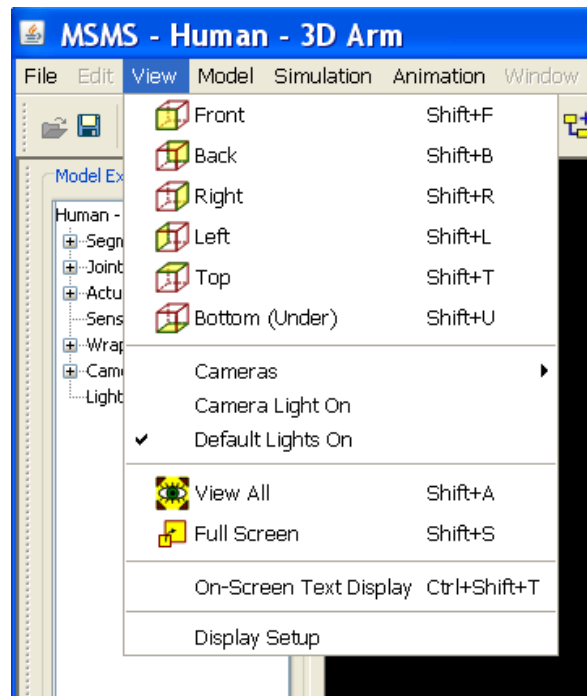


Models import models from SIMM and SolidWorks

3.9. Exit

This command closes the currently open model and exits MSMS.

4. View Menu



4.1. Points of View

4.1.1. Front

This command shows the Front view. Keyboard shortcut: Shift + F. The Front view is the Y-Z plane projection of the model with Y axis pointing upwards, and Z axis pointing to the left of the screen.

4.1.2. Back

This command shows the Back view. Keyboard shortcut: Shift + B. The Back view is the Y-Z plane projection of the model with Y axis pointing upwards, and Z axis pointing to the right of the screen.

4.1.3. Right

This command shows the Right view. Keyboard shortcut: Shift + R. The Right view is the Y-X plane projection of the model with Y axis pointing upwards, and X axis pointing to the right of the screen.

4.1.4. Left

This command shows the Left view. Keyboard shortcut: Shift + L. The Left view is the Y-X plane projection of the model with Y axis pointing upwards, and X axis pointing to the left of the screen.

4.1.5. Top

This command shows the Top view. Keyboard shortcut: Shift + T. The Top view is the X-Z plane projection of the model with X axis pointing upwards, and Z axis pointing to the right of the screen.

4.1.6. Bottom (Under)

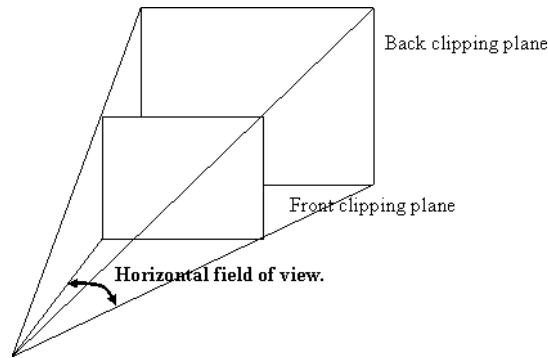
This command shows the Bottom view. Keyboard shortcut: Shift + U. The Bottom view is the X-Z plane projection of the model with X axis pointing upwards, and Z axis pointing to the left of the screen.

4.2. Cameras

All models include a default camera for viewing. The user however, can create any number of custom cameras for a model. All such cameras will be listed here and can be chosen as the active camera by the user. The view of a camera can be controlled by its parameters as listed in the following table.

| Camera Control | Description |
|-----------------|--|
| Position | Controlled by the XML tag <i><position></i> , with sub-tags <i><x></i> , <i><y></i> , and <i><z></i> , in meters. |
| Orientation | Controlled by the XML tag <i><orientation></i> , which specifies a 3x3 orientation matrix, with sub-tags <i><r1c1></i> , <i><r1c2></i> , <i><r1c3></i> , <i><r2c1></i> , <i><r2c2></i> , <i><r2c3></i> , <i><r3c1></i> , <i><r3c2></i> , and <i><r3c3></i> . |
| Projection Type | Controlled by the XML tag <i><projectionType></i> , with values <i>Perspective</i> or <i>Orthographic</i> . |
| Screen Scale | Controlled by the XML tag <i><screenScale></i> , with a dimensionless numeric value for the scale factor. |
| Field of View | Specifies the width of the camera's visibility. Controlled by the XML tag <i><fieldOfViewRad></i> , with a numeric value in radians. |
| Foreground | Specifies the foreground clipping point for the camera. Objects closer to the camera than this distance will not be visible. Controlled by the XML tag <i><frontClipDistance></i> , with a numeric value in meters. |
| Background | Specifies the background clipping point the camera. Objects farther from the camera than this distance will not be visible. Controlled by the XML tag <i><backClipDistance></i> , with a numeric value in meters. |

The figure below illustrates the combined effects of the field of view with the front and back clipping distance on the resulting visual frustum.



A default camera is always present in MSMS, with the characteristics indicated in the following table. The default camera cannot be modified by users, and only one camera can be active at a time, but users can employ the *Cameras* menu of the MSMS GUI to switch between the default camera and any other cameras that have been added to the world via the XML file.

| Default Camera Parameter | Value |
|------------------------------|-----------------------------|
| Position | (0, 0, 2.4142) meters |
| Orientation | (1, 0, 0; 0, 1, 0; 0, 0, 1) |
| Projection Type | Perspective |
| Screen Scale | 1.0 |
| Field of View | $\pi/4$ radians |
| Foreground Clipping Distance | 0.01 meters |
| Background Clipping Distance | 10.0 meters |

4.3. Camera Light On

This command toggles on and off the spotlight that is attached to the camera and always points in the same direction as the camera.

4.4. Default Lights On

This menu option toggles on and off the default lights which is a combination of one ambient light and two directional lights.

In addition to the camera and default lights, the user can add new lights. Lighting of the MSMS "world" can be accomplished by four basic types of light: (1) ambient, (2) directional, (3) point, and (4) spotlight. Descriptions of these light types and their various control options are provided in the following table.

| Lighting Type or Control | Description |
|--------------------------|--|
| Ambient | Models the light (called background light) reflected from other visual objects. For example, a room with no lights may not be pitch dark because light (from some external source) has entered the room, and bounced and reflected along room boundaries and other objects to reach the viewer's eye. Typically, only one <i>Ambient</i> light source is specified for any 3D environment. It is controlled by the XML tag <code><lightType></code> , using the value <i>Ambient</i> . |

| | |
|-----------------|--|
| Directional | Models a distant light source (like the sun), where all light rays emanate from the same direction, in parallel, and with equal intensity. When illuminated by <i>Directional</i> light, only parts of objects exposed to the rays will be illuminated. It is controlled by the XML tag <code><lightType></code> , using the value <i>Directional</i> . |
| Point | Models an Omni-directional light source (such as a light bulb), which emits light uniformly in all directions from a given position within the 3D environment. An <i>Attenuation</i> control is available, such that the farther away an object is from a <i>Point</i> light source, the less bright it will be. It is controlled by the XML tag <code><lightType></code> , using the value <i>Point</i> . |
| Spotlight | Models real-life light sources (such as flashlights or torches) that have reflectors or lenses. In addition to <i>Attenuation</i> , <i>Spotlight</i> sources provide <i>Direction</i> , <i>Concentration</i> , and <i>Spread Angle</i> controls. When specifying <i>Direction</i> , the <i>Spread Angle</i> control defines a cone in the direction of the light (such as a flash light that emits a cone of light in a particular direction). The <i>Concentration</i> control affects the intensity of light across a cross-section of the cone of light, or how much of the light is concentrated in the center of the cross-section, versus attenuating outwards. The higher the concentration, the more light is focused in the center. When concentration is zeroed, then spread of light will be uniform throughout the cross-section. Controlled by the XML tag <code><lightType></code> , using the value <i>Spot</i> . |
| Color | Specifies the color of the light source in terms of its Red, Blue and Green (RGB) components. Since RGB color manipulation may not be intuitive to all users, MSMS provides a color selector that can be used to select the color visually. Controlled by the XML tag <code><color></code> , with sub-tags <code><x></code> , <code><y></code> , and <code><z></code> controlling the RGB components, respectively. |
| Light On Status | Specifies whether the light source is on or off. Controlled by the XML tag <code><lightIsOn></code> , with values <i>true</i> or <i>false</i> . |
| Position | Specifies the position of the light source. Controlled by the XML tag <code><position></code> , with sub-tags <code><x></code> , <code><y></code> , and <code><z></code> , in meters. |
| Direction | Specifies the direction of the light, as a vector, that emanates from the light source at a specified position. Controlled by the XML tag <code><direction></code> , with sub-tags <code><x></code> , <code><y></code> , and <code><z></code> , in meters. |
| Attenuation | Specifies the attenuation of the light, as a function of distance. Defining the distance from the light source as "D", the three available attenuation options are: (1) No attenuation, where the light will not attenuate with distance, (2) Linear attenuation, where the light attenuates linearly with distance, as expressed by the equation: $\text{Intensity} = 1 / (1 + D)$, and (3) Quadratic attenuation, where the light attenuates as the square of distance, as expressed by the equation: $\text{Intensity} = 1 / (1 + D + D^2)$. Controlled by the XML tag <code><attenuation></code> , with values <i>None</i> , <i>Linear</i> , or <i>Quadratic</i> . |
| Concentration | Specifies the extent to which light is concentrated in the center of a cross-section of the radiated cone of light. A highly-concentrated light has most of its light rays near the cross-section center; the number of rays decreases with distance from the center. Controlled by the XML tag <code><concentration></code> , with values ranging from 0.0 (no concentration) to 1.0 (maximum concentration). |

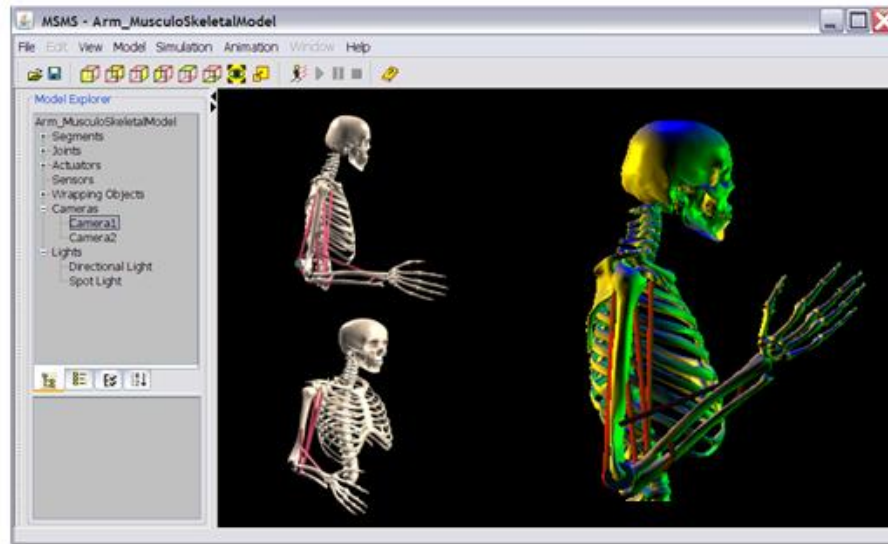
| | |
|--------------|---|
| Spread Angle | Specifies the angle between the specified direction of light and a ray along the outer-most edge of the resulting cone of light produced by the light source. Controlled by the XML tag <code><spreadAngleRad></code> , with values ranging from <i>0.0</i> to <i>90</i> , in degrees (not radians, tag name notwithstanding) . |
|--------------|---|

Control options for the various light sources are detailed in the following table (where N/A indicates that a given parameter is not used for the indicated light type). Note that in the XML `<color>` tags, the sub-tags `<x>`, `<y>`, and `<z>` actually control the RGB components of the light, respectively. Also, note that overall light intensity is not controllable directly, but is modified by the three color values (except for the *Spot* type, which offers an indirect intensity control via its *Concentration* parameter).

| Type | Color | On/Off | Direction | Position | Attenuation | Concentration | Spread |
|-------------|-------|--------|-----------|----------|-------------|---------------|--------|
| Ambient | Used | Used | N/A | N/A | N/A | N/A | N/A |
| Directional | Used | Used | Used | N/A | N/A | N/A | N/A |
| Point | Used | Used | N/A | Used | Used | N/A | N/A |
| Spotlight | Used | Used | Used | Used | Used | Used | Used |

A default light is attached to a default camera that is always present in MSMS, with the characteristics indicated in the following table. The characteristics of the light on the default camera cannot be modified by users, but users can employ the MSMS GUI to switch the default camera's light On or Off.

| Default Camera Light Characteristic | Value |
|-------------------------------------|------------------------------------|
| Type | Spotlight |
| Color | White |
| On/Off Status | On |
| Position | Same as default camera position |
| Orientation | Same as default camera orientation |
| Attenuation | None |
| Concentration | 64 |
| Spread Angle | PI / 2 radians |
| | |



4.5. View All

This menu option redirects the camera toward the model and moves it back until the whole model can be seen. This is useful when the camera is pointing to the wrong direction and the model is invisible.

4.6. Full Screen Mode

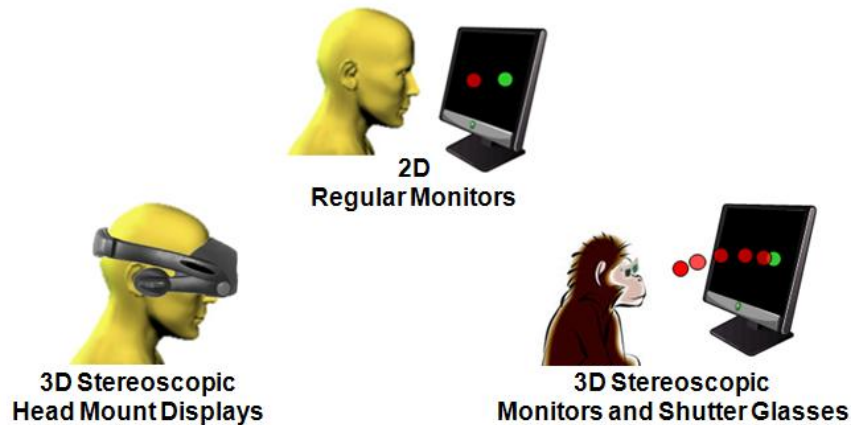
This menu option enlarges the 3D window to cover the entire screen. The keyboard shortcut for this command is Alt + Ctrl + F. You can use the same shortcut or press Esc to get back to the normal mode.

4.7. On-Screen Text Display

This menu option creates a billboard for text display in 3D window and prepares MSMS to receive the position of the billboard and the text from a simulation program. The simulation program (e.g Simulink) must use MSMS's feature commands to send these data to MSMS (see the section on Animation). To learn more on how to display text in 3D window, see Appendix E: Feature Commands.

4.8. Display Setup

Display setup allows you to setup various 2D and 3D displays for virtual reality simulations.

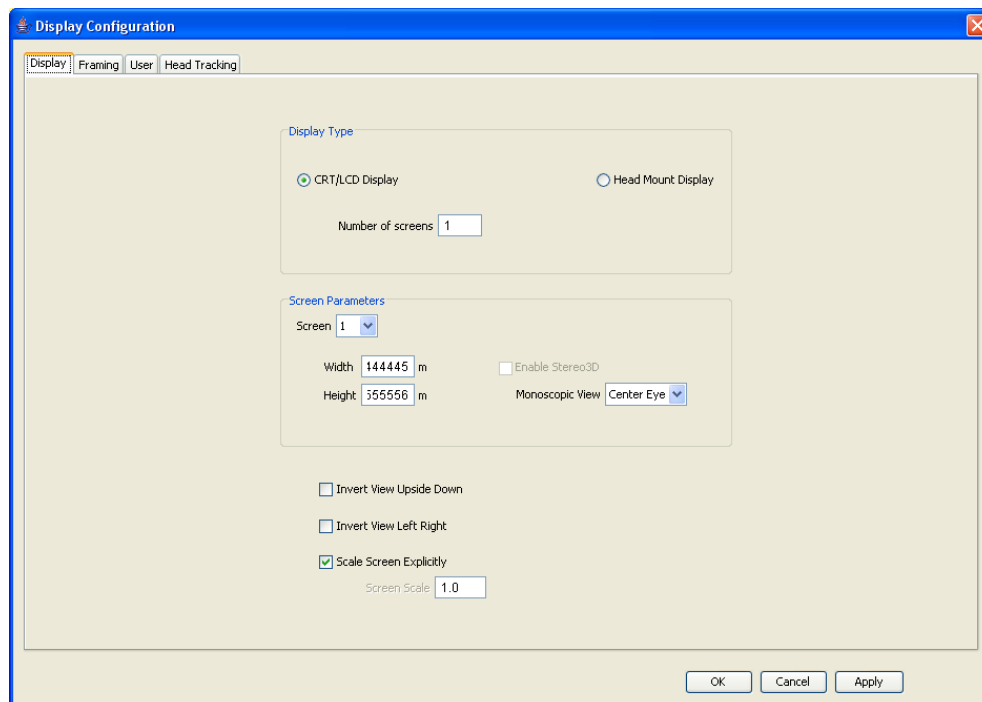


Display Setup includes four tabs:

4.8.1. Display

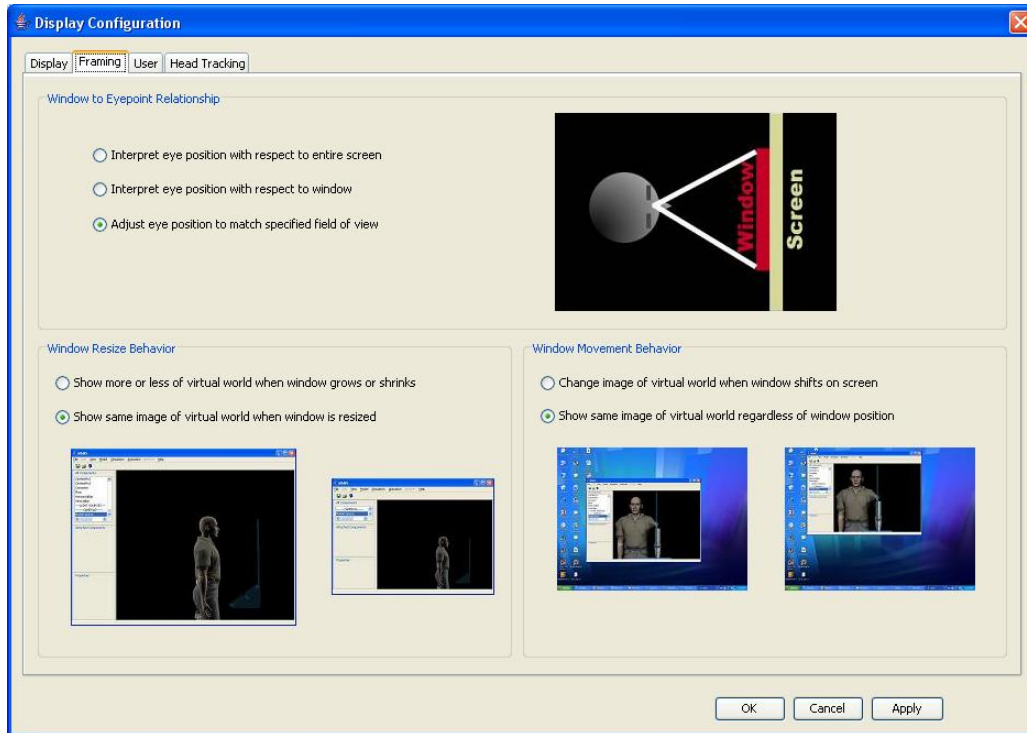
In the Display Tab the following can be edited:

1. The number of screens that the display contains. This would be required in case of head mount displays or multi-projection displays both of which have multiple screens and the user needs to configure each screen individually.
2. The monoscopic view needs to be set in some cases, for example, a head mount display that gets its left and right feed from two different video channels. In that case, you cannot get stereo vision by just enabling stereo 3D. Instead you need to set the monoscopic view of one screen to "Left Eye" and the other screen to "Right Eye".



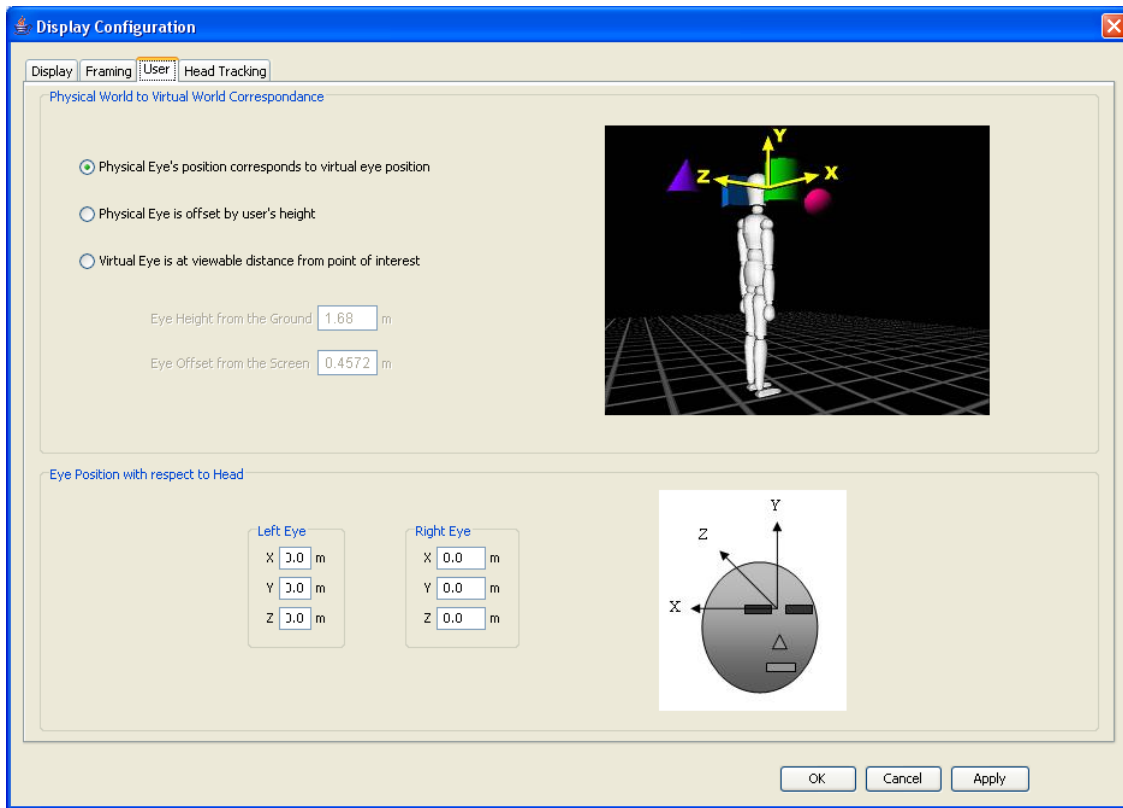
4.8.2. Framing

The Framing Tab controls aspects specific to how the MSMS window affects the view as seen by the user. Based upon the settings, the view frustum gets modified. All the settings shown here are the initial settings.



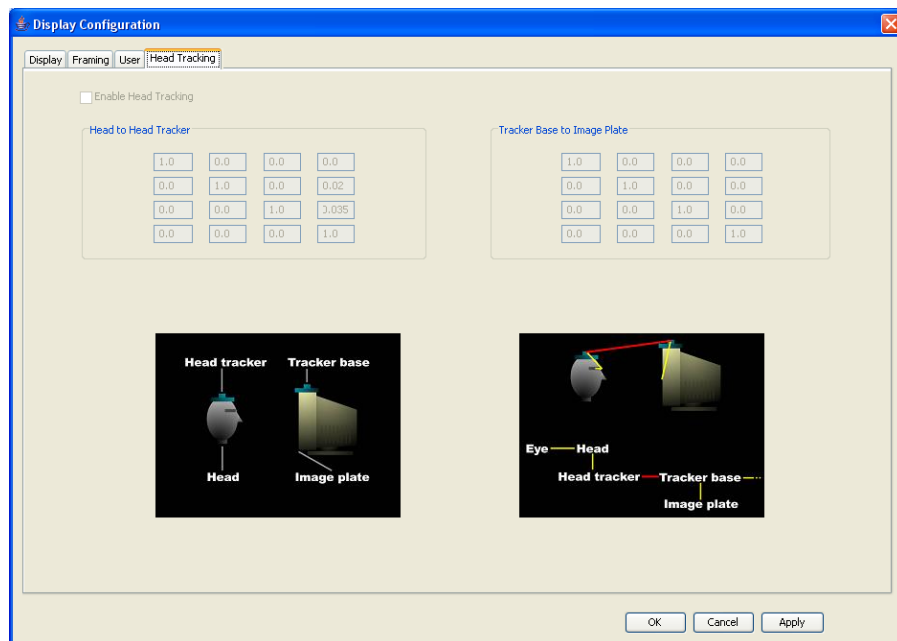
4.8.3. User

The User tab is used to control settings that are user dependant, e.g. the position of eyes. When the physical to virtual eye correspondence is set to "Virtual Eye is at viewable distance from point of interest", the eye position will need to be specified with respect to the screen and not with respect to the head. The title text "Eye Position with respect to Head" will change to "Eye Position with respect to Image Plate" to reflect this.



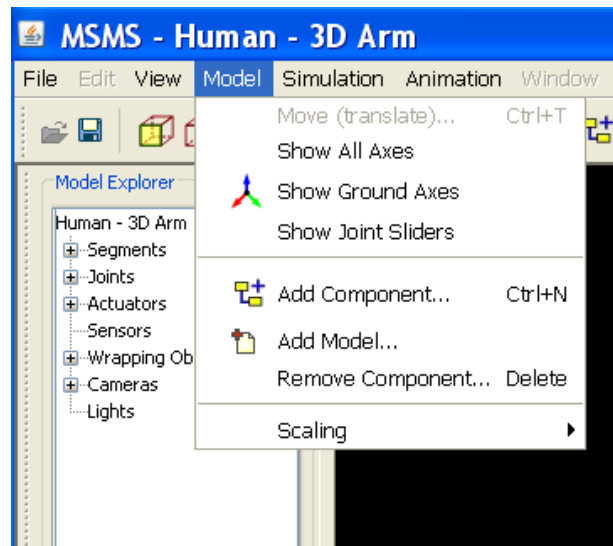
4.8.4. Head Tracking

The Head Tracking tab allows you to enable head tracking. The matrices shown are for head tracking with head mount display. When the user chooses "CRT/LCD Display" in the Screen tab, different set of matrices will be shown.



5. Model Menu

Includes tools for model manipulation and editing.



5.1. Move (translate)

Currently, this item is not used.

5.2. Show All Axes

Shows/hides the local co-ordinate axes of all components.

5.3. Show Ground Axes

Shows/hides Ground reference frame.

5.4. Show Joint Sliders

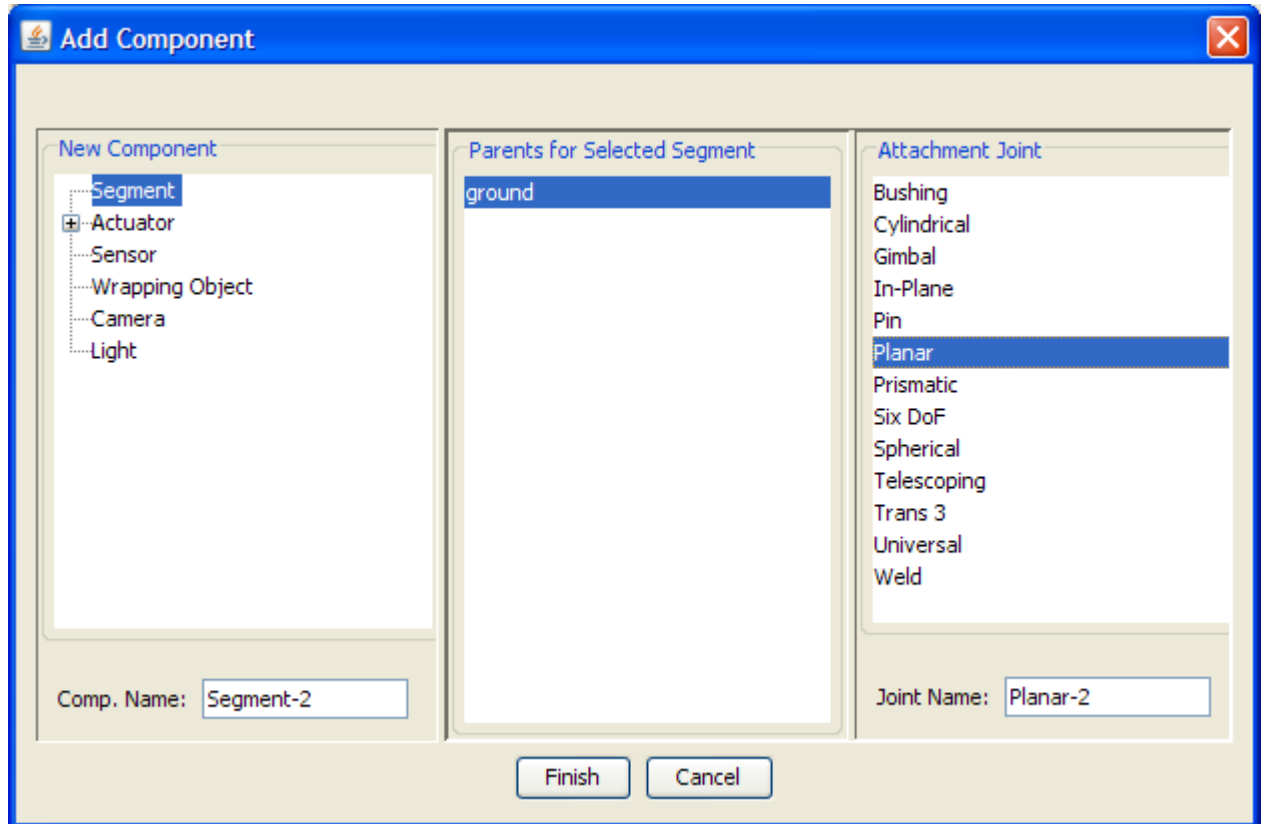
Shows/hides joint sliders for all degrees of freedom in the model. The sliders can be used to move individual degrees of freedom in the model.

5.5. Add Component

Brings up the add component wizard that allows you to add components such as segments, actuators, sensors, wrapping objects, cameras, and lights. Joints are automatically created when you create a segment. Once a component is inserted, it will show up in the model explorer as the currently selected component.

The newly created component may belong to one of the Human, Prosthesis, or World Category. If you attach the new component to an existing component, it will inherit its parent's category by default. If you are adding a component to a new blank model, it

will assume Human Category by default. If this is not desirable, open the component's GUI after its creation and change its category. Please note that if *you have created a segment-joint pair, you must change the category in both so that they belong to the same category.*



5.5.1. Segment Appearance

To provide the maximum flexibility in designing the appearance of the segments, MSMS allows the users to add any number of shapes to represent a segment. These shapes can be individually added, removed, edited, and positioned on a segment.

By default, when you add a segment to your model, it does not have any shapes attached to it. Therefore, you can see the newly added segment in the Model Explorer but not in the 3D model window. You can design the appearance of your segment by adding image to your segment. To add an image, double click on the new component name and edit its image properties in the image tab.

MSMS allow you to add two types of shapes to your segments: primitives and 3D meshes.

Primitives: Can be used in segments that have relatively simple appearance. The supported primitives include cylinder, sphere, box, hemisphere, and capped cylinder. Because MSMS allows you to add any number of shapes to each segment, you can combine multiple primitives to build more complex shapes, if necessary. Once a primitive shape is added to a segment, all of its properties such as material properties,

its position and orientation with respect to the segment, and its scaling parameters can be edited independent of any other shape attached to that segment. This provides the maximum flexibility and enables the users to build segments with the desired appearance even with the use of simple primitives.

3D Meshes: Can model any 3D shape and can be used to model the appearance of segments with complex shapes such as human body segments, bones, and prosthetic limbs. To add a 3D mesh to a segment, add a new image and select Mesh from the 3D Shape drop down menu. Then import the 3D mesh by clicking on the Add button that will allow you to browse for the 3D mesh files. For complex shapes made of multiple 3D meshes, you can either add more images to the segment each containing one 3D mesh or you may add multiple 3D meshes to one image. In the later case, any editing to the image will apply to all of the 3D meshes attached to it.

The 3D meshes may be in one of the many 3D file formats but the one that is completely supported by the MSMS is OBJ file format. So, before importing to MSMS, you must first convert your 3D meshes to OBJ file format, if needed.

Before adding 3D meshes to your segment, you must have created or obtained these 3D meshes. You can obtain your 3D shapes by purchasing them from the 3D model vendors, by finding and downloading them free of charge from the Internet, or by creating them yourself in a 3D modeling software such as MAYA, 3ds Max, Blender, etc. These software tools allow you to create new 3D shapes, customize them by editing their shape and texture, and perform file format conversion, when necessary.

5.6. Add Model

Using this command, one can add an entire human, prosthesis, or world model to the current model. For example, a user can remove one prosthesis in a model and add a different one. This allows you to quickly combine a human model with various prosthesis and world models.

5.7. Remove Component

This command will remove the currently selected component from a model. You will see a warning if the removal affects other attached components.

5.8. Scaling

This menu option allows you to scale either a segment or the whole model. Scaling for a segment allows you to scale it along a vector defined by two points (usually the proximal joint and one of the distal joints). Scaling the whole model, once implemented, will allow you to scale the model along x, y, and z axes.

5.9. Model Info

This menu option allows you to obtain component ID numbers and the joint sequence numbers.

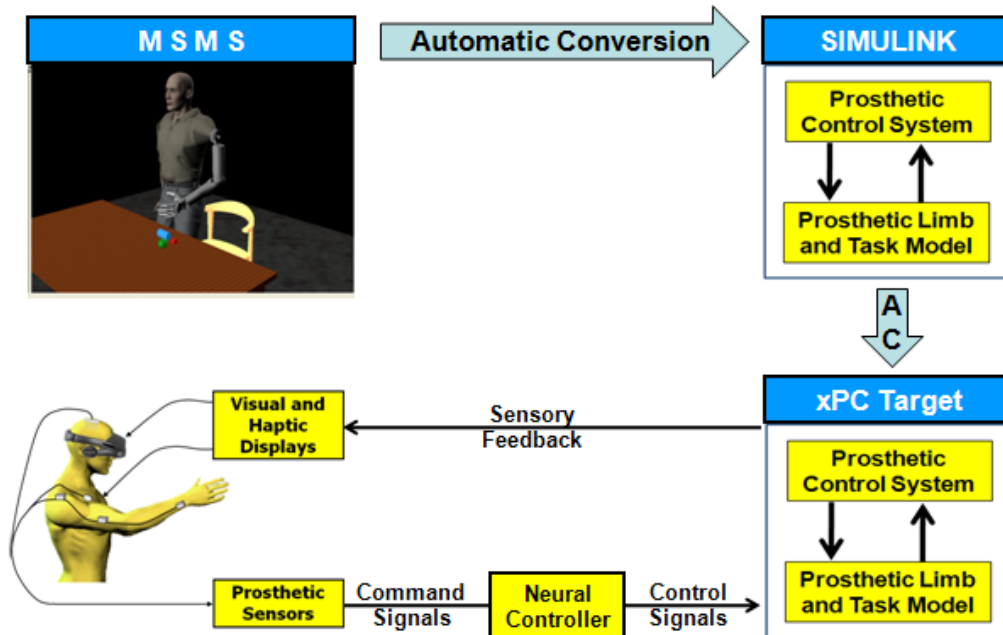
When developing a simulation program in Simulink or other environments to send Feature commands to MSMS, the user needs to know the component IDs and the joint

sequence numbers. These numbers are available in the model XML files and can be obtained by opening them in a text editor and searching for the IDs and the sequence numbers. But this process is time consuming and inconvenient.

The Model Info menu options allow you to display the component IDs in the Model Explorer or write model IDs or the joint sequence numbers into a text file in an easy to read tabular format.

6. Simulation Menu

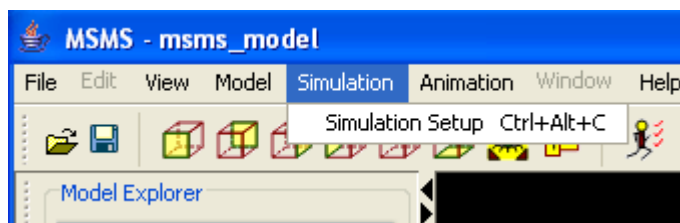
MSMS models can be automatically converted into Simulink for physics-based simulations of the model's behavior in response to control inputs and external forces. Another automatic conversion using Matlab's xPC Target toolbox allows you to run the Simulink model in a real-time PC for virtual reality simulations with the subject in the loop.



The Simulation Setup allows you to setup the simulation configuration before exporting it to Simulink.

6.1. Simulation setup

This option allows you to setup the simulation before saving it to Simulink. Once the simulation is setup, it can be saved into a Simulink model from file menu.



Each branch of the simulation tree can be setup independently to be fixed during simulation, animated by motion data, or simulated dynamically according to the laws of physics. This provides flexibility in setting up versatile simulation environments.

While the simulation is running, the animation menu in MSMS can be configured to receive and animate the resulting motion (See Animation Menu).

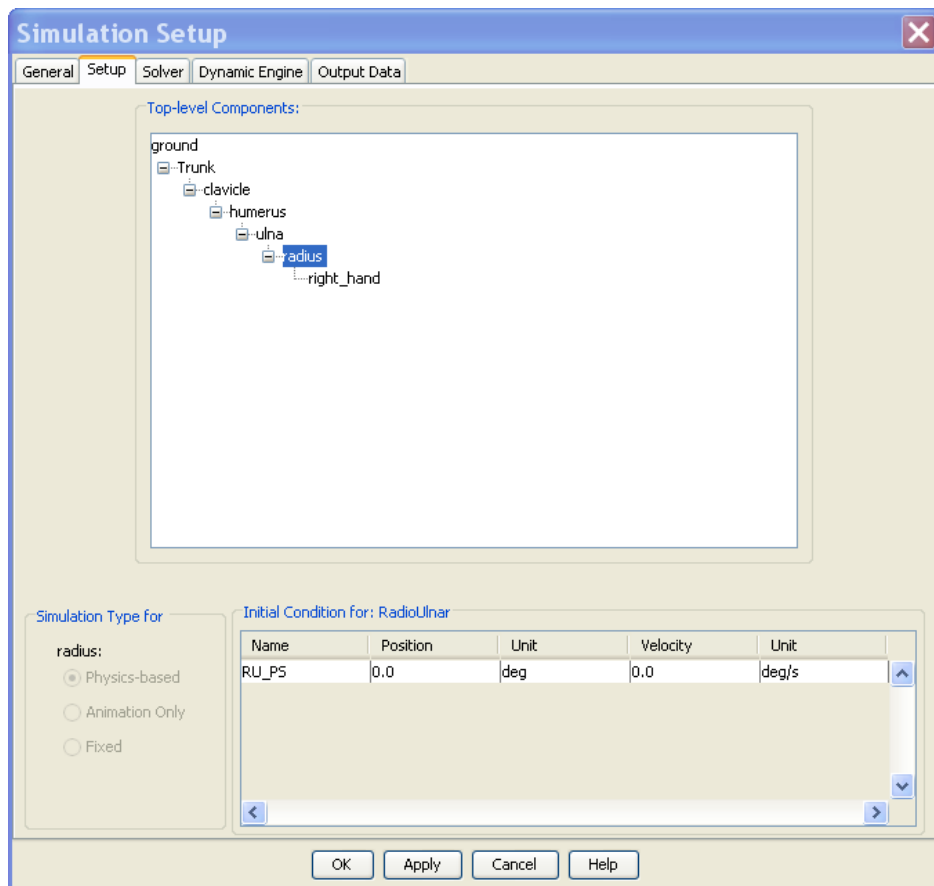
The configuration window contains five tabs which are explained below.

6.1.1. General

Here, you can set the gravity vector for physics-based simulation. The realistic default vector of $[0 \ -9.81 \ 0]$ is rarely changed, if ever.

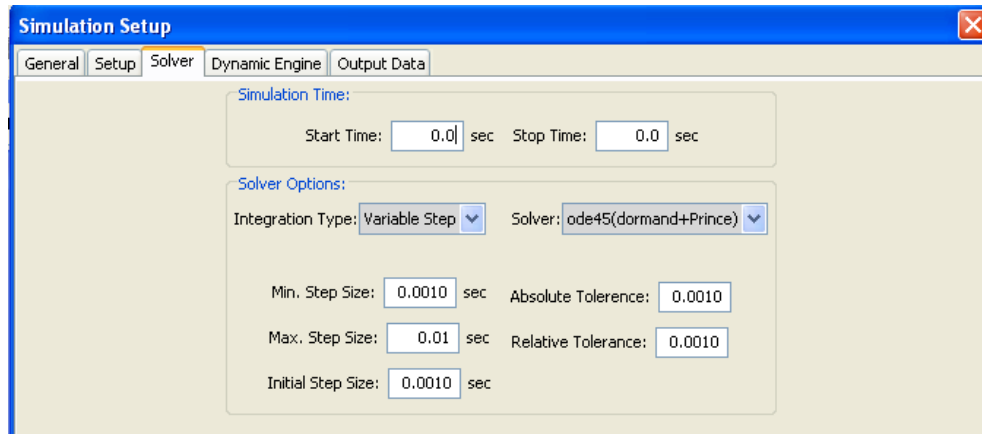
6.1.2. Setup

In the setup tab, you can see the tree structure view of the model, setup initial conditions for each degree of freedom. The grayed out simulation type, once implemented, will allow you to specify the simulation type for each branch of the model tree.



6.1.3. Solver

Here, the simulation time can be chosen. The solver options allows you to select the numerical solver that Simulink uses to solve the simulation's dynamic equations. These parameters greatly affect the performance of the solver and the behavior of the simulation. For more information, please refer to the 'solver parameters' section of Simulink help pages.



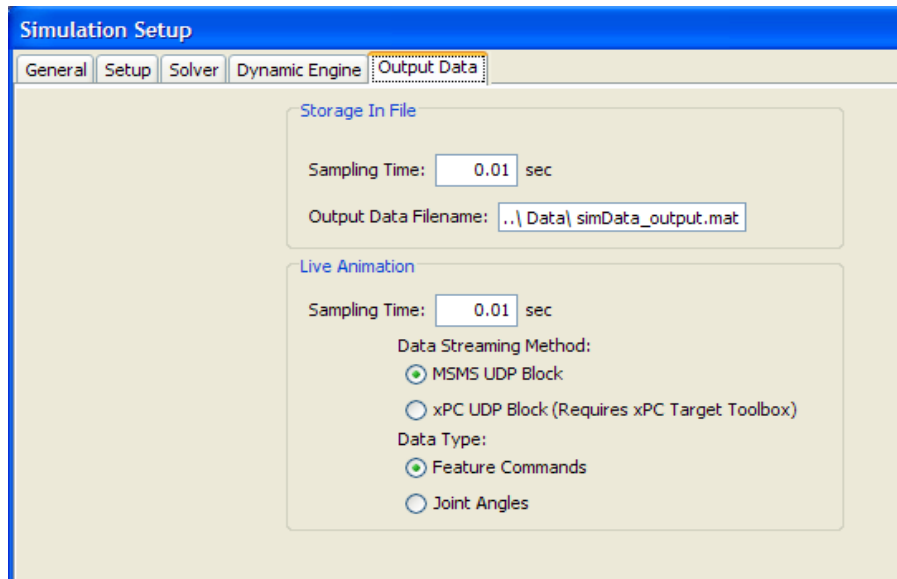
6.1.4. Dynamic Engine

Here, the dynamic engine can be chosen. However, currently only MATLAB's SimMechanics is supported.

6.1.5. Output Data

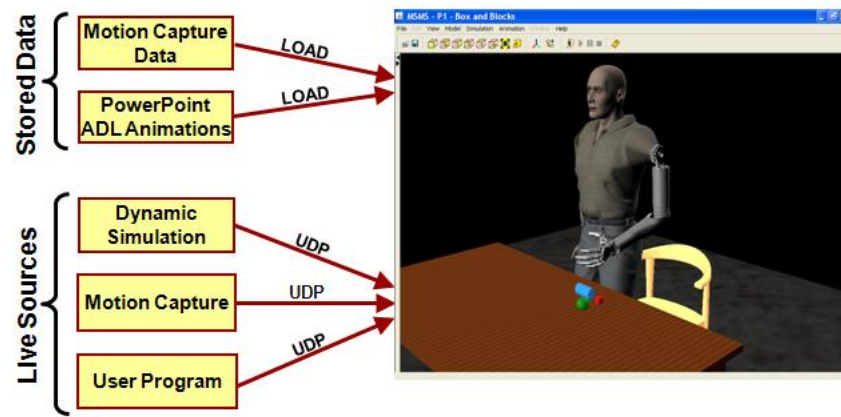
Here, you select the type and frequency of simulation data that must be stored in output files. There are two data streaming methods in MSMS. In other words, there are two methods via which Simulink can send animation data to MSMS. They are: MSMS UDP Block, and xPC UDP Block. You can choose the desired method here. Use xPC UDP Block if you have xPC Target toolbox. But if you don't have this toolbox, you can use MSMS UDP Block.

The animation data can be sent in one of the two formats: Feature Commands and Joint Angles. The Joint Angles format can be used to send only the joint motions but Feature Commands is a more comprehensive format and can be used to send not only the joint motions but also commands to control the VR simulations such as sound playback, drawing trajectories, changing the color or size of objects, etc. For more see Appendix E: Feature Commands

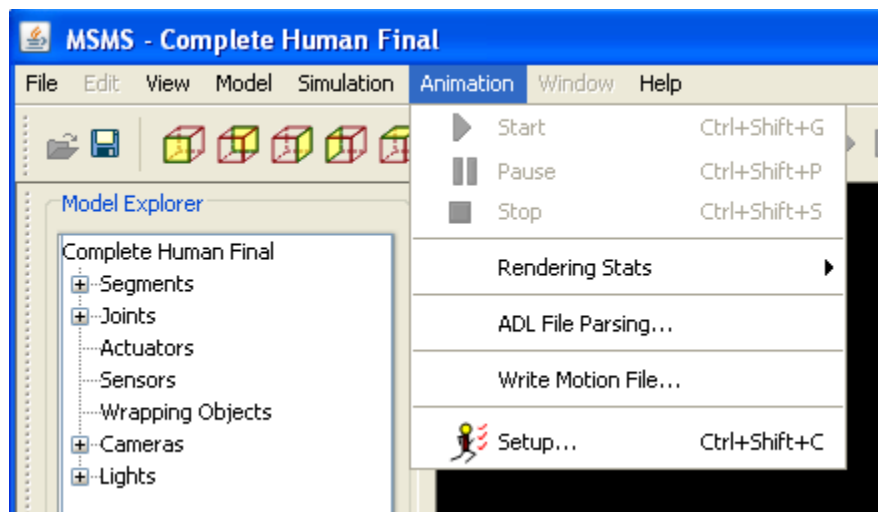


7. Animation Menu

MSMS Models and viewing angles (cameras) can be animated using recorded data or motion data streamed in real-time from variety of live sources.



In Animation menu you can setup and run off-line and live animations of MSMS models.



7.1. Start

This command causes MSMS to animate the model. It will be active after the animation is Setup.

7.2. Pause

This command pauses animation. It will be active after the animation is Setup.

7.3. Stop

This command stops animation. It will be active after the animation is Setup.

7.4. Rendering Stats

This command displays the MSMS rendering times in the status bar. You can use this feature to see how fast MSMS and your visualization PC can render a specific model. This evaluation can help you take actions such as simplifying the model or acquiring faster video cards if necessary.

7.5. ADL File Parsing...

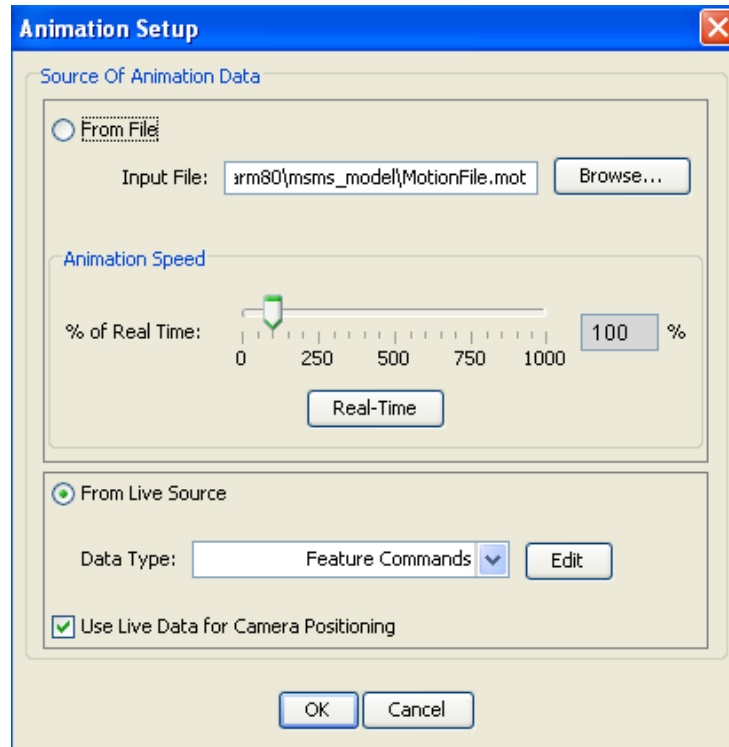
This command allows you to convert an ADL animation sequence built in Microsoft PowerPoint to the msm motion file format that can be animated in MSMS. For more details, see Appendix F: ADL Animations.

7.6. Write Motion File...

This command allows MSMS to create a msm motion file with single frame of data corresponding to the current posture of the model in MSMS. There are two potential uses for these motion files. First, because these files have the correct motion file format, they can be used as starting point to manually create new motion files. Second, motion files created in the start and end postures of a movement can be used to create primitive motion files that interpolate the frames in between these two postures. A Matlab script is written to perform this function. For more details, see Appendix F: ADL Animations.

7.7. Setup

Here, you can setup the animation by selecting the source of animation data and its parameters. This setup enables MSMS to load motion data from a file or prepares it to receive the motion data from a live stream. After this setup the Start, Pause, and Stop buttons will be activated.



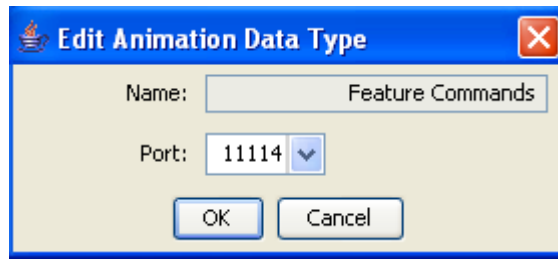
From File: The animation data will be read from a motion file in one of three formats: SIMM's motion file (.mot), MSMS's motion file (.msm), or motion data stored in Matlab's binary format (.mat). For more see Appendix D: Motion File Formats.

From Live Source: The animation data will be received from a live source in real-time such as dynamic simulation running in Simulink or a motion capture system. Using the drop-down list select the type of simulation data that will be sent by the live source. The choices are:

- Feature Commands
- Ordered Joint Angles
- Ordered Joint Angles - APL

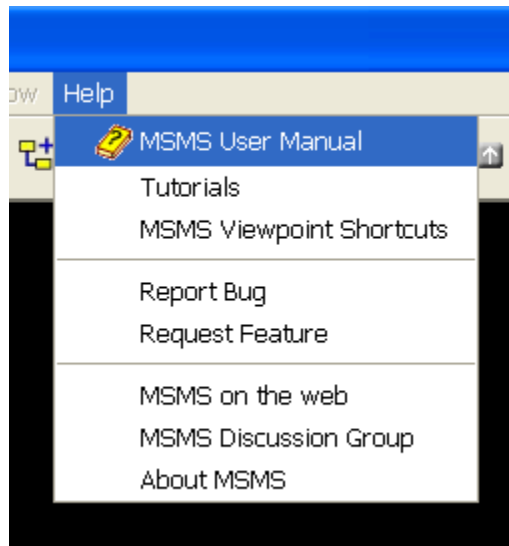
Feature Commands allows you to control model properties beyond its motion such as object sizes and color etc. Ordered Joint Angles, only sends the joint motion data to animate the model. Ordered Joint Angles - APL is specifically designed for users at APL and must not be used by others.

By clicking on the Edit button, you can select the UDP port number through which the live source and MSMS will send and receive animation data, respectively.



Check "Use Live Data for Camera Positioning" if the live source will send data to control the position and orientation of the camera.

8. Help Menu



8.1. MSMS User Manual

This command opens this document.

8.2. Tutorials

This command opens the tutorial folders in windows explorer. The tutorials are designed to introduce the main features of MSMS. Each tutorial folder contains the instructions for tutorial, the required files to do the tutorials, and the final results after successful completion of the tutorial.

8.3. MSMS Viewpoint Shortcuts

This command shows the list of keyboard shortcuts for controlling the view of the model in 3D window.

8.4. Report Bug

This command opens your default email application and allows you to report MSMS bugs and defects to the MSMS development team.

8.5. Request Feature

This command opens your default email application and allows you to request new MSMS features.

8.6. MSMS on the Web

This command opens the MSMS web page.

8.7. MSMS Discussion Group

This command displays information about MSMS discussion group MSMS-L.

8.8. About MSMS

This command displays info about MSMS and its developers.

9. Appendix A: MSMS Files and Directory Structure

MSMS uses a directory structure to store and trace the models and their corresponding simulation data. The directory structure creates snapshots of the complete model system, including the entire current contents of the XML files specifying the modeled system. This directory structure consists of folders for the Model Library and the Simulation Library.

9.1. Model Library folder

This folder stores the generally available XML descriptions of different human, prosthesis, and world models, “views,” and images (3D and other images stored in various formats, such as obj, jpg, etc.) in separate subfolders, respectively. The Views folder contains saved template “view” files, which contain data related only to how the model is viewed and do not describe any aspect of the model itself. The view file is described in greater detail below. The Images folder contains the primary library of images used by the models. Each image is given its own subdirectory that contains all the files necessary to produce the image (e.g., an image subdirectory might include an .obj file, an .mtl file and several .jpg files). The Model Library is used to build subject models.

9.2. Simulations folder

This folder stores subject-specific directories. Within each subject-specific directory are one or more configurations that each contains a model possibly along with data required to run a simulation using this model and data that is generated by the simulations. Each Simulation Configuration folder contain either a completely unique model or a variation of a single model that differs from other configurations only by variations in the initial and boundary conditions and/or simulation settings.

An example of the latter is that a specific model could be configured so that the prosthetic left arm uses physics-based simulation while the right sound arm is animated by data from a motion capture system. The same model could also be configured to use physics-based simulation for both arms. These variations could be configured within MSMS GUI and exported automatically to multiple simulation configuration folders.

Each simulation configuration folder stores a complete description of the model used to generate the data. This is important because models evolve over time, but the exact model that was used in the simulation is required to properly analyze, animate, or interpret the data.

The role of the Images directory under a particular simulation configuration model directory is to store images particular to the model that are not found in the main image library. The MSMS will search for images referenced in the model files by looking first in the model directory’s Images directory (and subdirectories), if any. If the image file is not found, the MSMS will then search the main image library and subdirectories.

9.3. Workspace.xml

This is a file maintained separate from this directory structure to store user/installation specific data. This data includes screen size, screen viewing preferences, and other items such as which directories were last visited. This file is located in the user’s home directory (located under the “Documents and Settings” directory for Windows users).

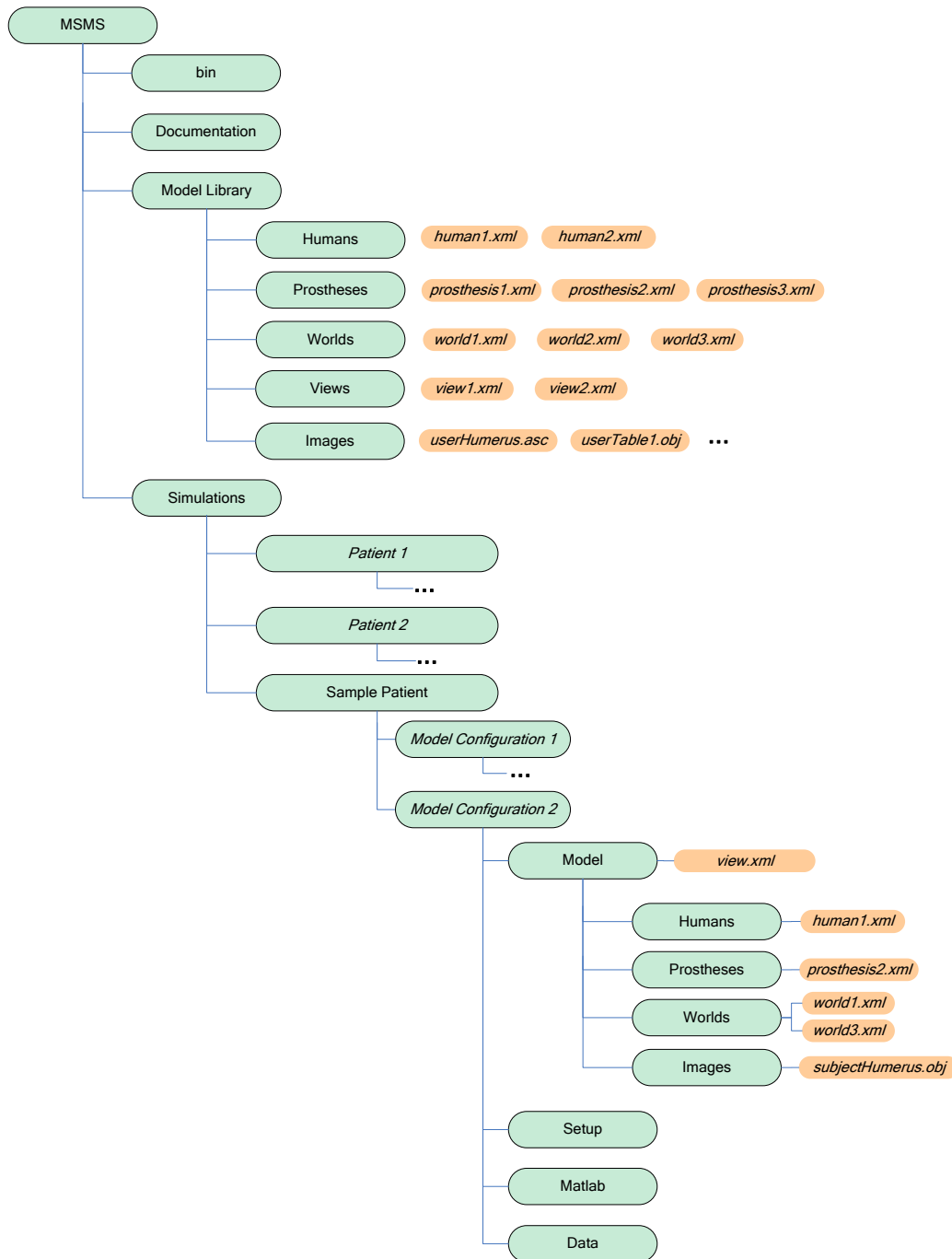
9.4. View.xml

the view file contains data that only pertains to the viewing of the model. This data does not define any aspect of the model itself. Therefore, changes to this file do not affect any simulation data that may be generated. Examples of the contents of the view file are

- 1) Flag indicating if the ground axes are displayed
- 2) Background color
- 3) Camera descriptions, including position and orientation
- 4) Lighting information

Such view data may also be saved as a template in the Model Library folder's Views directory so that a user may easily use the same view for several models.

Model specific parameters related to view (e.g. display of joint axes) cannot be included in a View file because it can be used with various models that each have different components.



The figure above shows the structure of these directories in the context of the entire MSMS distribution directory structure. All directory and file names are fixed except for those in italics font.

The steps for creating a new model are given below, along with how MSMS utilizes the directory structure:

- 1) From the MSMS GUI, the user selects the files from the Model Library folders that will comprise the desired model. The files will be zero or more XML files from each of the directories: Humans, Protheses, Worlds and Views.

- 2) The MSMS creates a new Simulation Configuration directory under a particular subject's directory that contains only the selected model and view files. The name of the Simulation Configuration directory is chosen by the user (if this is for a new subject, the entire subject-specific structure will be created).
- 3) Using the MSMS GUI, the user then tailors the model to the particular situation that is required to run the desired simulation.
- 4) At any point, the user can request the MSMS to save the model. Under the current Simulation Configuration, the MSMS creates a simulation setup file (simulationSetup.xml) and saves any changes to the model and view. If a view file had not been selected when the model was created, a view file is created that pertains specifically to the current model.
- 5) Once the model is tailored to the users satisfaction, the user may select the Save Simulation menu item under the File menu. This will create the mdl file or other simulation driver file that the user may run to control the animation.
- 6) The user, inside or outside the MSMS depending on the situation, runs the simulation program, generating animation data. This data may either directly animate the model as it is generated or the generated data may be saved to a file in the "Data" directory for future use.
- 7) Once simulation data is generated, any modifications to the model invalidate both that data and the simulation driver file. This does not apply to changes to the View file because such changes do not affect the model itself. This integrity of the relationship between model and data is maintained in the following ways:
 - a. If the model is modified and the user attempts to run an animation with existing data, the user will be notified that the data does not pertain to the current model.
 - b. If the model is modified and the user attempts to save the model, the user will be given two choices:
 - i. The model may be saved back to the current Simulation Configuration directory, but all simulation data including the simulation driver file will be deleted.
 - ii. The model may be saved to a new Simulation Configuration directory (which will not yet have any simulation data).

10. Appendix B: Component GUIs

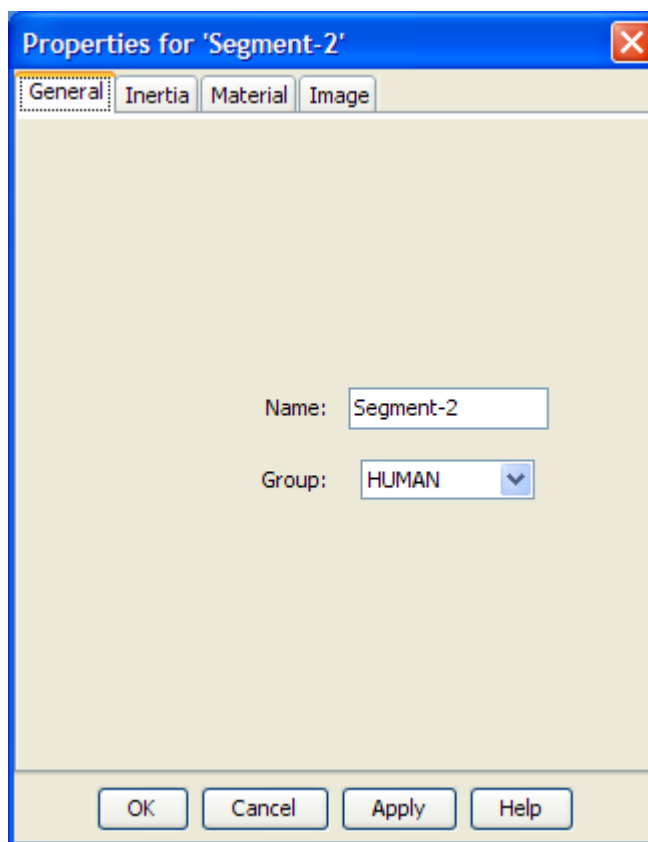
10.1. Segment

The properties window for a segment includes 5 tabs. Each tab enables the user to view and edit a number of related properties of the segment.

10.1.1. General

Name: A unique name for the segment such as: Humerus.

Group: All the components in a model are divided into three groups: human, prosthesis, and world. You can set a component's group using this drop-down list.



10.1.2. Inertia

Mass: Mass of the segment.

Center of Mass Offset: Center of mass offset for the segment as a vector in the Cartesian X, Y, Z directions.

Inertia Matrix: The inertia matrix of the segment has the form:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

Properties for 'Head'

General **Inertia** Material Joint Centers Image

Mass

0926315 kg

Center of Mass Offse

X 1781814 m

Y 0000006 m

Z 10777E-6 m

Inertia

kg*m^2

| | | |
|-------------|-------------|-------------|
| 1.97991E-5 | 6.57608E-6 | -4.64541E-8 |
| 6.57608E-6 | 3.15739E-4 | -7.31053E-9 |
| -4.64541E-8 | -7.31053E-9 | 3.25293E-4 |

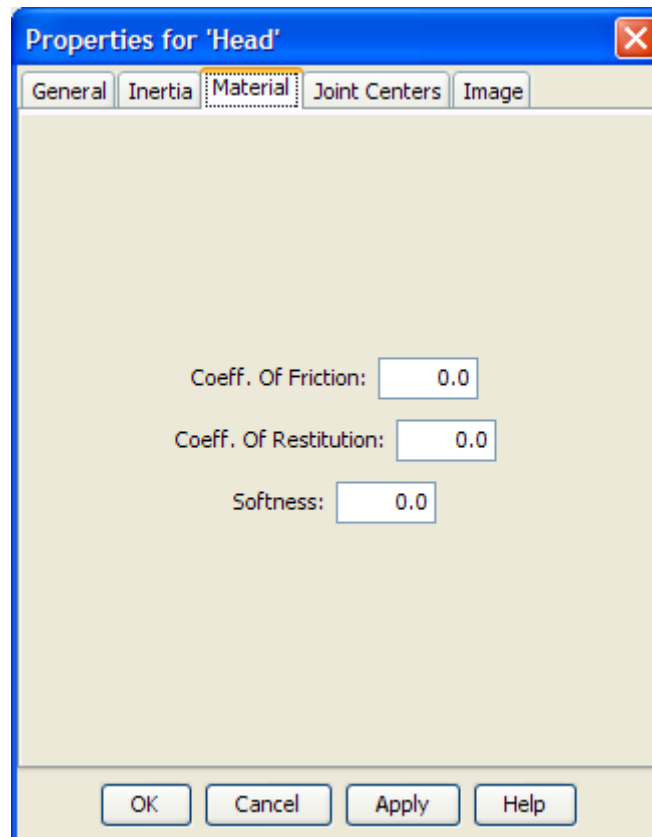
OK Cancel Apply Help

10.1.3. Material

Coefficient of Friction: Coefficient of friction for segments is not yet implemented.

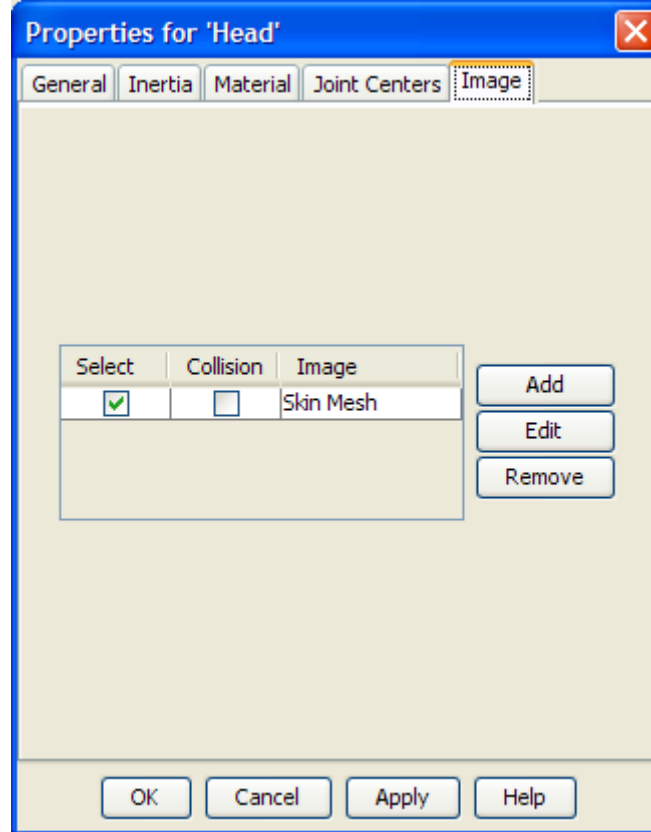
Coefficient of Restitution: The coefficient of restitution is a value between 0 and 1 and determines the elasticity of the segment. A value of 0 means that the segment is completely elastic (bouncy), while a value of 1 means that the segment is completely inelastic (sticky).

Softness: Softness is a value between 0 and 1 and determines how deformable the segment is.



10.1.4. Image

A list of images attached to the segment is shown.



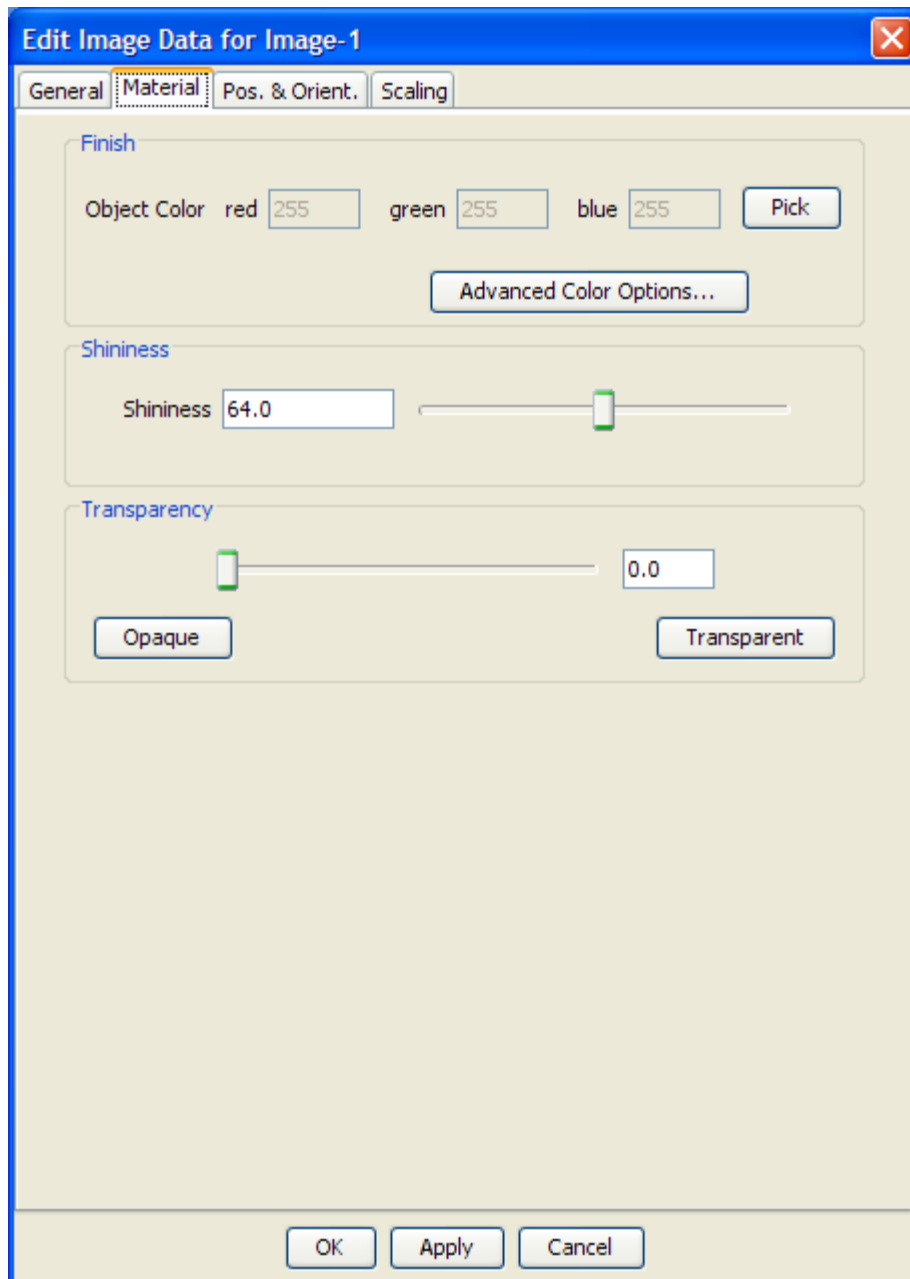
Select: This checkbox toggles on and off the visibility of the image in the 3D pane.

Collision: This checkbox toggles on and off whether image is included in collision detection and handling.

Add: This command adds an image to the segment. The image's properties can be set via a 4 tab window.

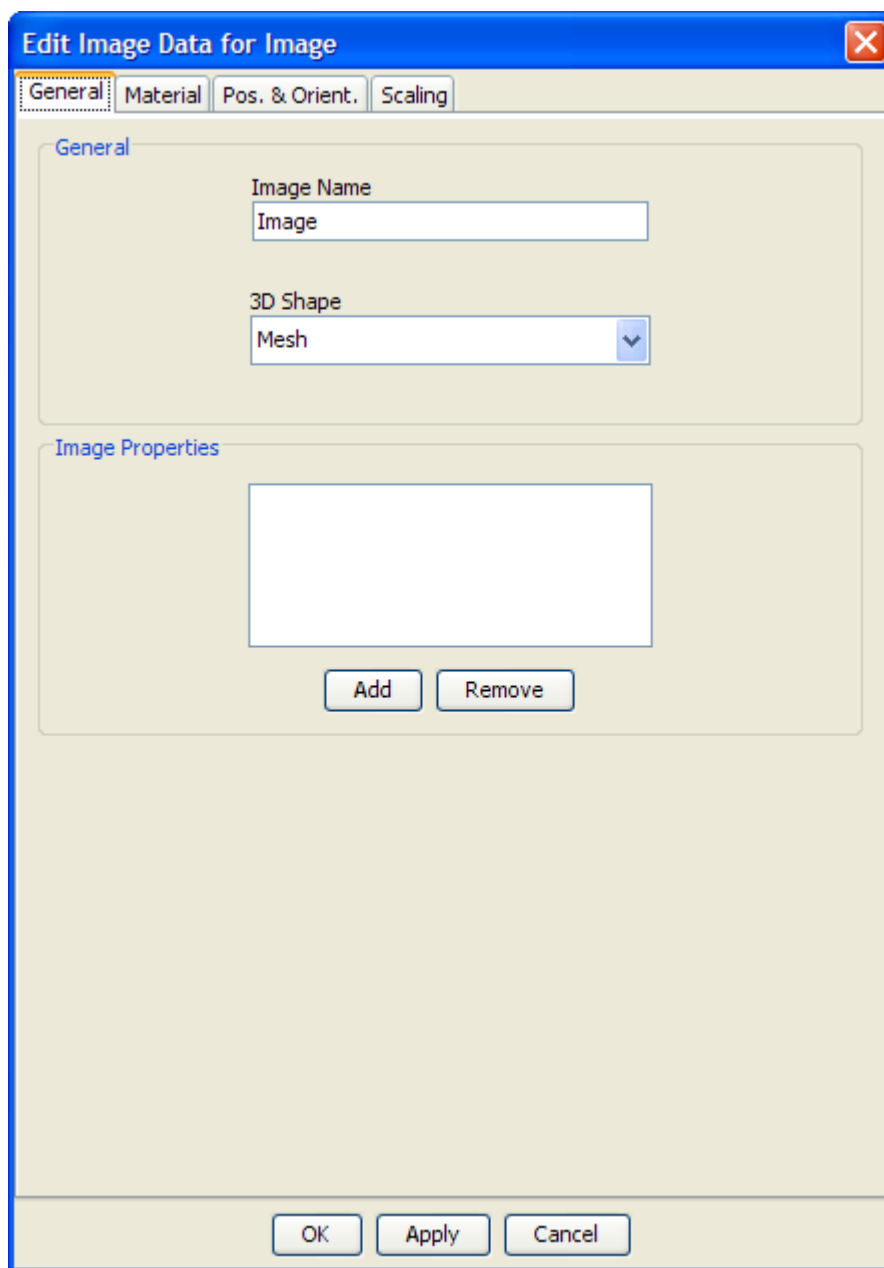
General: An image name can be chosen here. The name does not need to be unique. For the shape of a 3D image, there are five primitive choices and a general mesh choice. The primitives are: Cylinder, Sphere, Box, Hemisphere, and Capped Cylinder.

Material: Here, the object's appearance can be set by choosing: Object color, Ambient, Diffuse, Specular, Shininess, and Transparency. You can select an object's color by clicking on "Pick". The color selected will be applied to all color types for the object, i.e., ambient, diffuse, and specular. Therefore, it will be approximately the color of the object in all light types. Advanced users can select the color for each individual color type by clicking on "Advanced Color Options". For more information, please refer to the description of light color in MSMS in the Lights section (12).



Position & Orientation: Here, the image can be translated along and rotated about any of the x, y, and z axes.

Scaling: Here, the image can be scaled along any of x, y, and z axes, or uniformly along all axes.

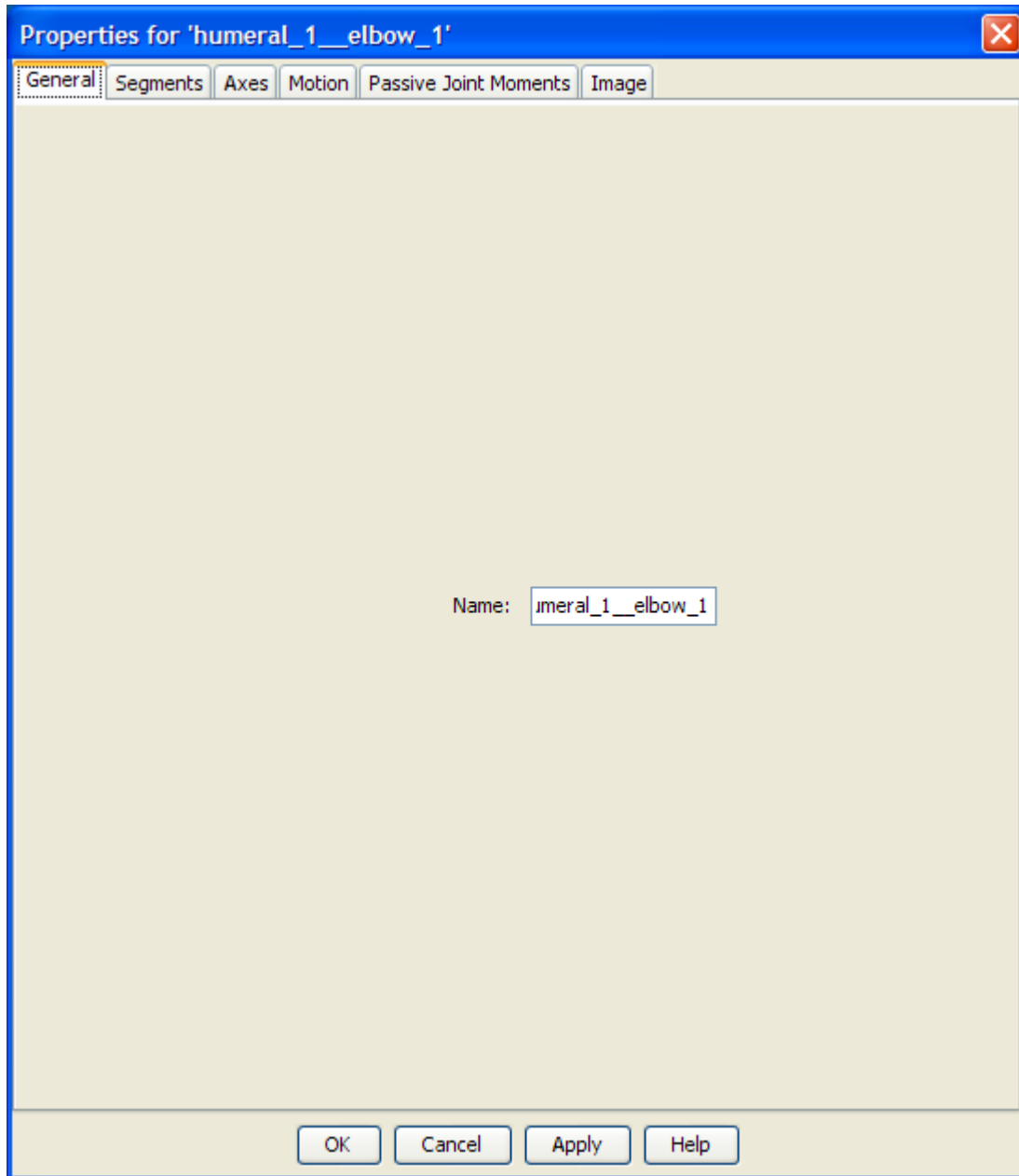


Edit: Choose an image in the list and select this button to edit its properties.

Remove: Choose an image in the list and select this button to remove it from the segment.

10.2. Joint

The properties window for a joint includes 6 tabs. Each tab enables the user to view and edit a number of related properties of the joint.



10.2.1. General

Name: A unique name for the joint such as: humeral elbow.

10.2.2. Segments

Properties for 'humeral_1__elbow_1'

General Segments Axes Motion Passive Joint Moments Image

Proximal Segment

Name:

Joint Center Offset

X: m

Y: m

Z: m

Distal Orientation (Rx->Ry->Rz)

X: deg

Y: deg

Z: deg

Distal Segment

Name:

Joint Center Offset

X: m

Y: m

Z: m

OK Cancel Apply Help

Proximal Segment

Name: Sets the proximal segment for this joint.

Joint Center Offset: Sets the joint center offset.

Distal Orientation: Sets the distal orientation as angles about the X, Y, and Z axes.

Distal Segment

Name: Sets the distal segment for this joint.

Joint Center Offset: Sets the joint center offset.

10.2.3. Axes

The screenshot shows a software window titled "Properties for 'humeral_1__elbow_1'". It has a tabbed interface with "General", "Segments", "Axes" (selected), "Motion", "Passive Joint Moments", and "Image".

Inside the "Axes" tab, there is a "Joint Type:" dropdown menu set to "Pin/Revolute". Below this are two sections:

- Translational Axes:** A box containing the text "None."
- Rotational Axes:** A table with two columns: "Name" and "Axis Vector (meters)".

| Name | Axis Vector (meters) |
|------------------|----------------------|
| 1. al-1--elbow-1 | 0474361 .998874 0.0 |

At the bottom of the window are four buttons: "OK", "Cancel", "Apply", and "Help".

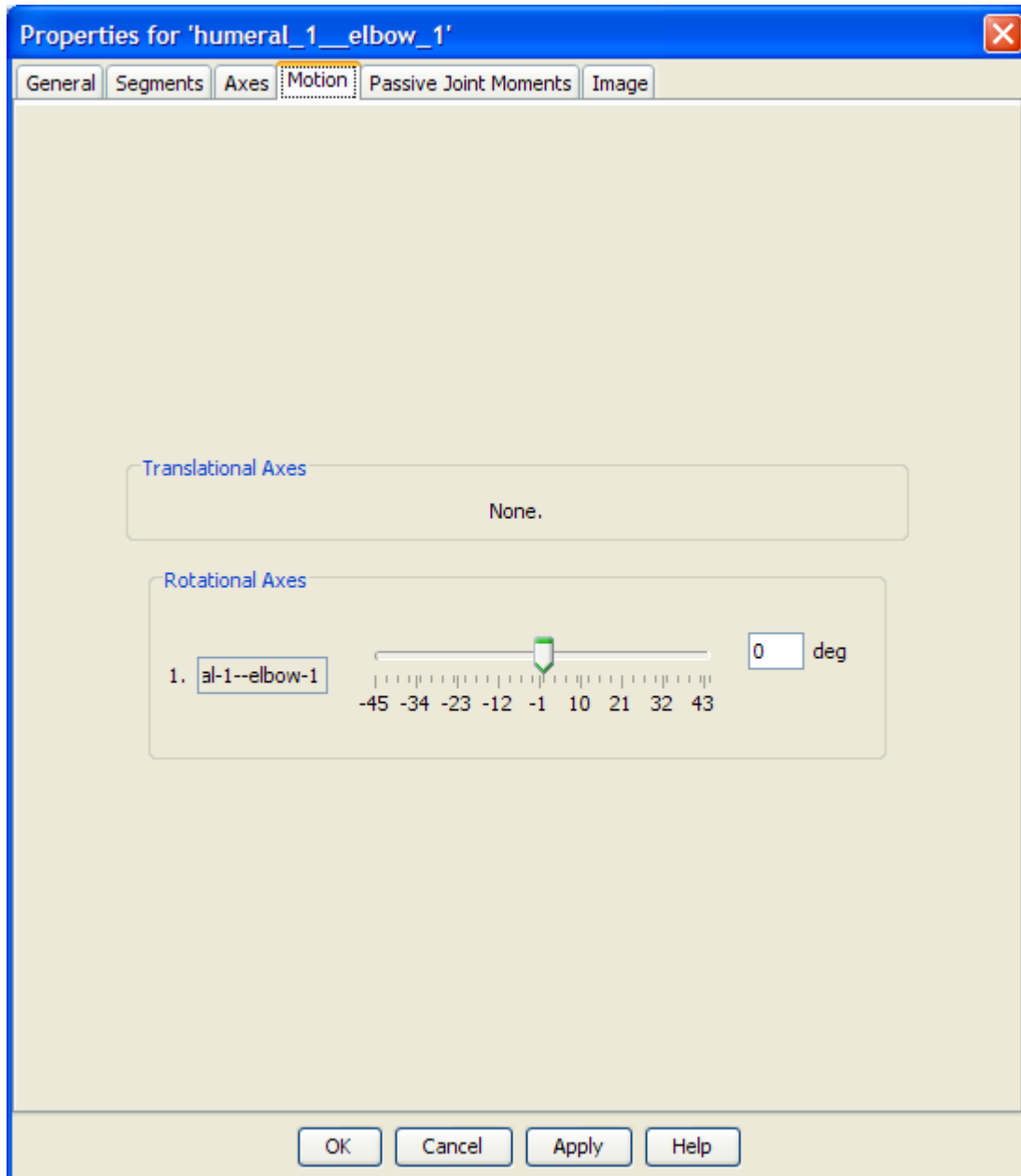
Joint Type

Sets the type of joint to one of: Pin/Revolute, Slider/Prismatic, Cylindrical, Universal/Hooke's, Planar, Gimbal, Ball/Spherical, Bearing, Bushing, Free (6 DOF), Telescoping, In-Plane, Weld.

Translational Axes: Sets the name and direction of the translational axes of the joint. Click the numbered button next to the name field to set the minimum, maximum and default values of the translational axis.

Rotational Axes: Sets the name and direction of the rotational axes of the joint. Click the numbered button next to the name field to set the minimum, maximum and default values of the rotational axis.

10.2.4. Motion



Translational Axes: The slider and numerical field allow you to change the value of the axis and hence move any segments distal segments attached to the joint.

Rotational Axes: The slider and numerical field allow you to change the value of the axis and hence rotate any segments distal segments attached to the joint.

Passive Joint Moments

The screenshot shows a software window titled "Properties for 'humeral_1__elbow_1'". It has a tabbed interface with "Passive Joint Moments" selected. The window contains the following fields and controls:

- DOF Name:** A dropdown menu showing "humeral-1--e..." with a downward arrow.
- Damping Factor:** A text input field containing the value "0".
- Low Limit:** A section with an "Activate" checkbox (unchecked). Below it are three fields: "Value (deg):" with "-45", "Torque Function:" with a dropdown set to "Linear", and "Coefficient:" with "1".
- High Limit:** A section with an "Activate" checkbox (unchecked). Below it are three fields: "Value (deg):" with "45", "Torque Function:" with a dropdown set to "Linear", and "Coefficient:" with "1".

At the bottom of the window are four buttons: "OK", "Cancel", "Apply", and "Help".

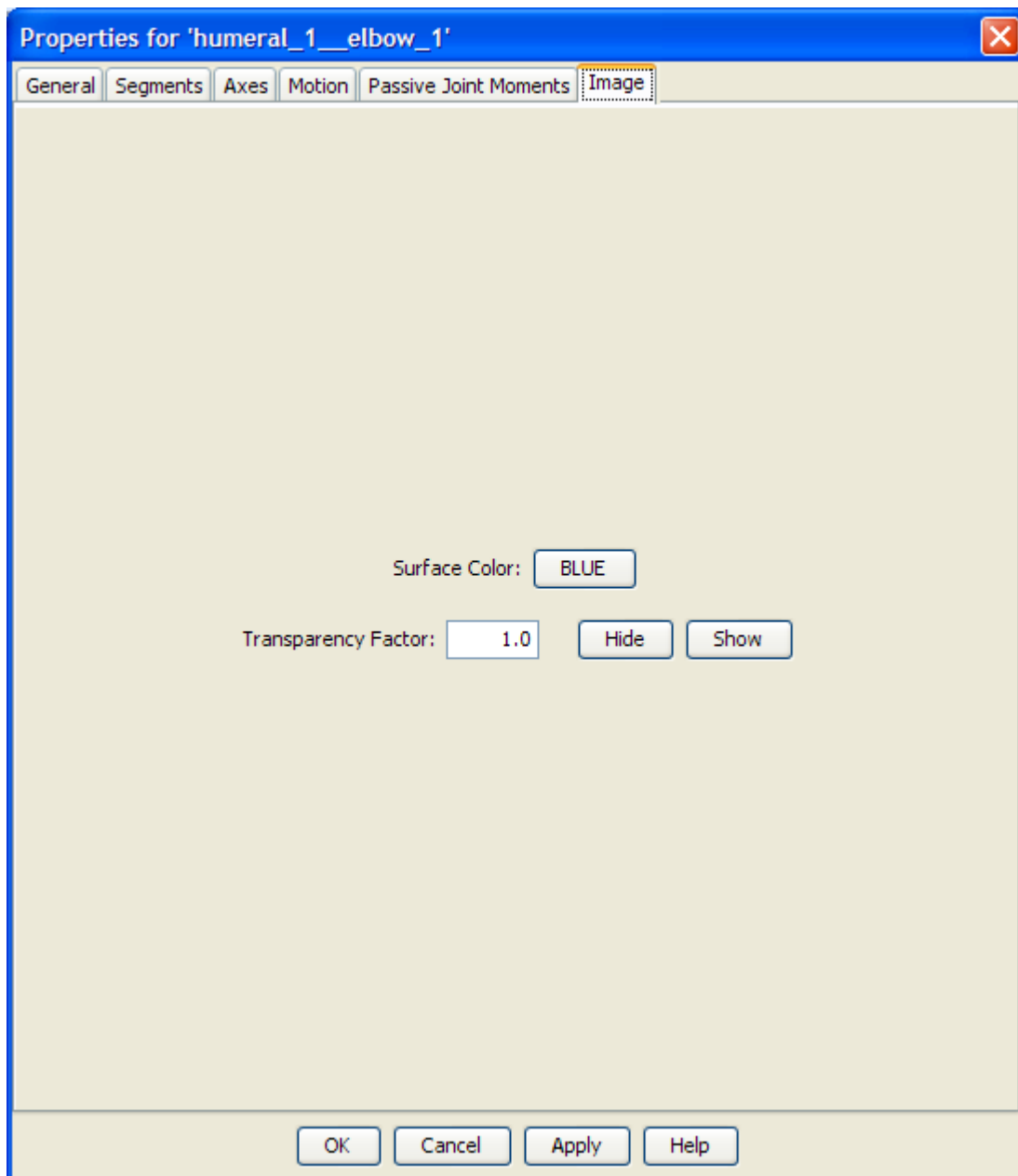
DOF Name: Set a name for the DOF.

Damping Factor: Set a damping factor.

Low Limit: Set a lower limit on the DOF along with a value, torque function, and coefficient.

High Limit: Set a higher limit on the DOF along with a value, torque function, and coefficient.

10.2.5. Image



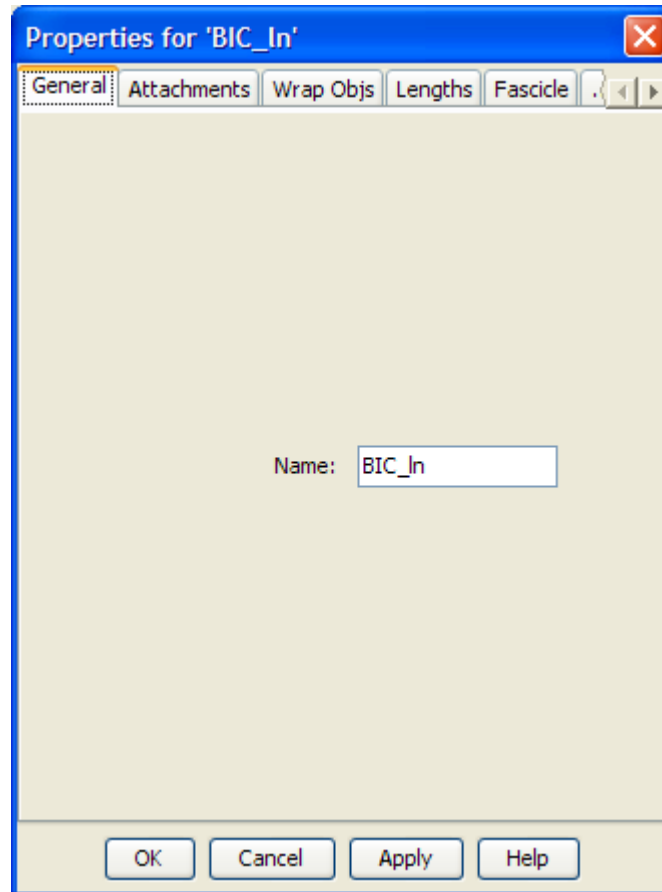
Surface Color: You can set the surface color of the joint blue to make it clearly visible in the 3D pane.

Transparency: You can set the transparency of the joint. Choose a value of 0 to hide it, or a value of 1 to make it completely opaque.

10.3. Muscle

10.3.1. General

Name: A unique name for the muscle component.



10.3.2. Attachments

Origin: Determines the origin segment of the muscle and its attachment point.

Insertion: Determines the insertion segment of the muscle and its attachment point.

The insertion and end points of a muscle tendon can be edited graphically. The split point represents the actual tendon length. The arrow keys also change the tendon length when the split point control is active. Also, the status bar on the bottom of the window shows the tendon length.

The screenshot shows a software window titled "Properties for 'BIC_In'". It has a tabbed interface with "General", "Attachments", "Wrap Objs", "Lengths", and "Fascicle". The "Attachments" tab is active. The window is divided into two main sections: "Origin" and "Insertion".

Origin Section:

- Segment: A dropdown menu showing "clavicle".
- Attach. Point:
 - X: -0.04989 m
 - Y: -0.00429 m
 - Z: 0.11332 m

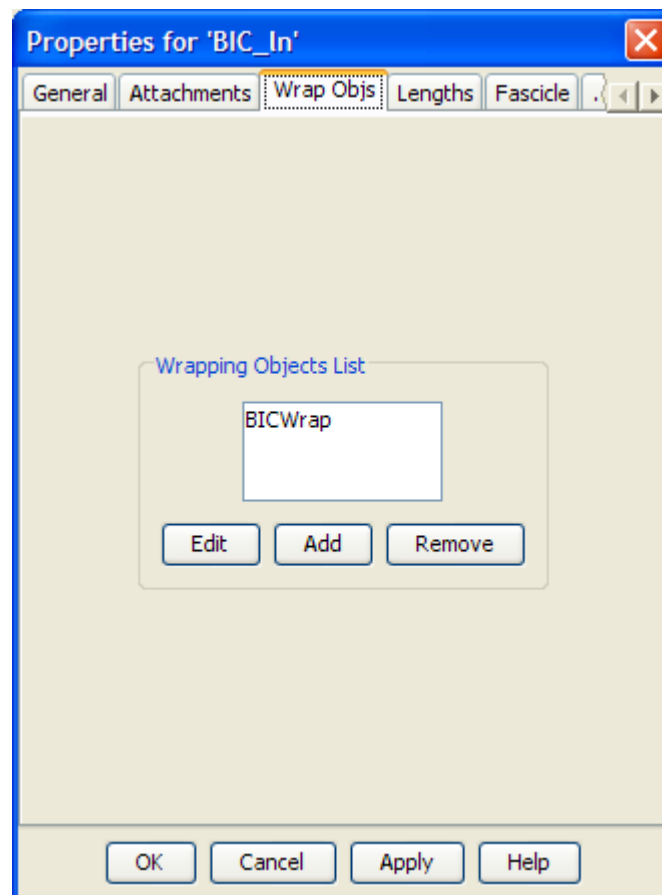
Insertion Section:

- Segment: A dropdown menu showing "radius".
- Attach. Point:
 - X: 0.01364 m
 - Y: -0.03203 m
 - Z: 8.7E-4 m

At the bottom of the window are four buttons: "OK", "Cancel", "Apply", and "Help".

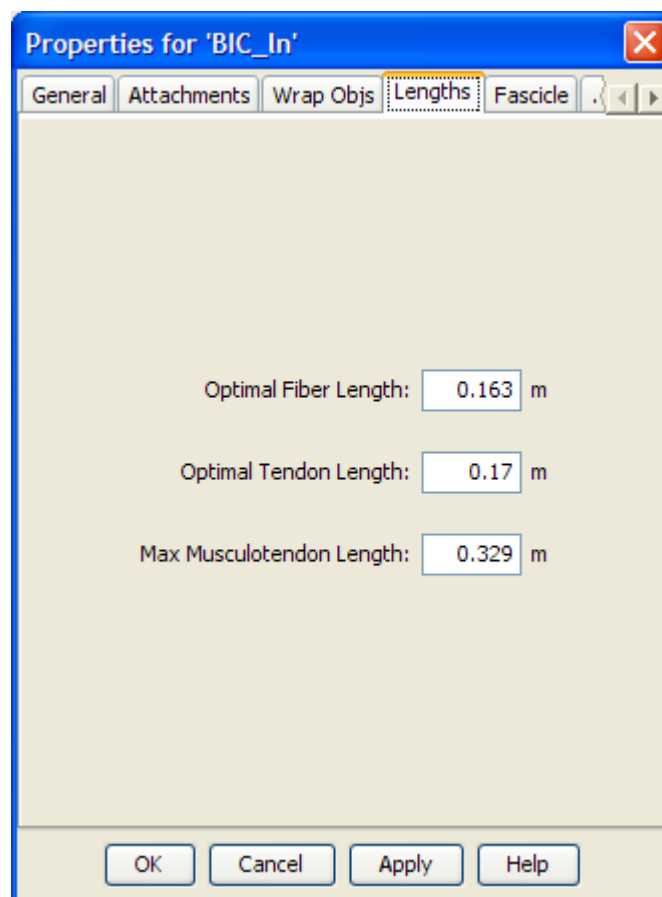
10.3.3. Wrap Objects

Here, the muscle's wrapping objects can be added, removed and edited.



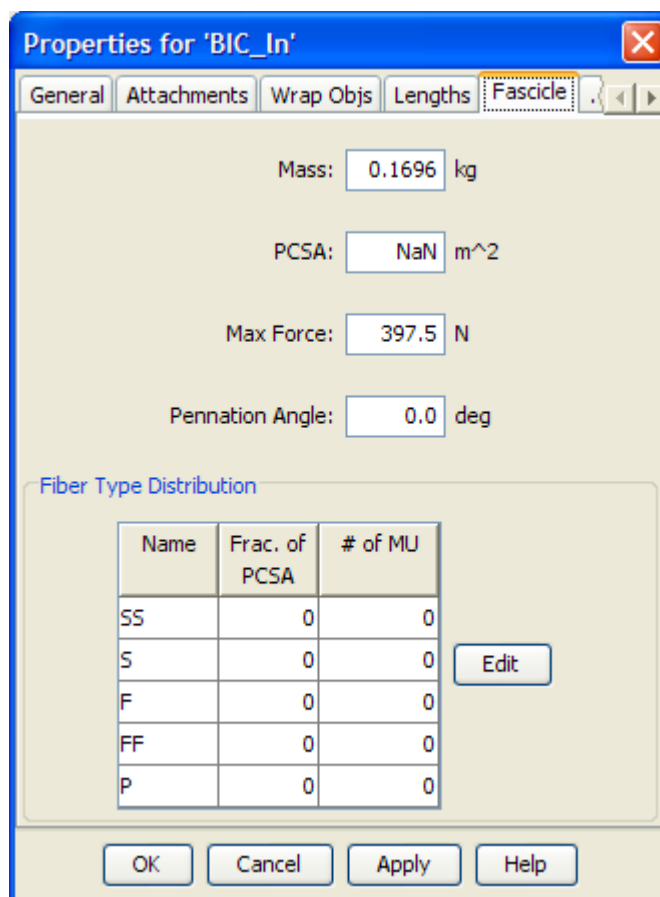
10.3.4. Lengths

The optimal fiber length, optimal tendon length, and the max musculotendon lengths can be set.



10.3.5. Fascicle

Here, the mass, PCSA, max force, and pennation angle can be set. Also, the fiber type distribution can be viewed and edited.



The image shows a software dialog box titled "Properties for 'BIC_In'". It has a tabbed interface with tabs for "General", "Attachments", "Wrap Objs", "Lengths", and "Fascicle". The "Fascicle" tab is currently selected. The dialog contains several input fields for muscle properties: "Mass" (0.1696 kg), "PCSA" (NaN m^2), "Max Force" (397.5 N), and "Pennation Angle" (0.0 deg). Below these is a section titled "Fiber Type Distribution" which contains a table with three columns: "Name", "Frac. of PCSA", and "# of MU". The table lists five fiber types: SS, S, F, FF, and P, each with a fraction of 0 and a count of 0. To the right of the table is an "Edit" button. At the bottom of the dialog are four buttons: "OK", "Cancel", "Apply", and "Help".

Mass: 0.1696 kg

PCSA: NaN m²

Max Force: 397.5 N

Pennation Angle: 0.0 deg

Fiber Type Distribution

| Name | Frac. of PCSA | # of MU |
|------|---------------|---------|
| SS | 0 | 0 |
| S | 0 | 0 |
| F | 0 | 0 |
| FF | 0 | 0 |
| P | 0 | 0 |

Edit

OK Cancel Apply Help

Clicking on the edit button opens the edit window. Here, the fiber types and their properties are displayed, and can be edited.

| Name | Recruitment Rank | V0.5 (L0/s) | f0.5 (pps) | fmin (f0.5) | fmax (f0.5) | Comments |
|------|------------------|-------------|------------|-------------|-------------|-----------------|
| SS | 1 | -0.515 | 8.5 | 0.5 | 2 | "super-slow" sc |
| S | 2 | -1.212 | 20 | 0.5 | 2 | generic slow-tw |
| F | 3 | -2.555 | 34 | 0.5 | 2 | generic fast-tw |
| FF | 4 | -1.055 | 15 | 0.5 | 2 | generic super f |
| P | 5 | -3.555 | 31 | 0.5 | 2 | user-defined fi |

Clicking on the Generic Coefficients edit button opens the corresponding window. Here, the generic coefficient values are displayed and can be edited.

Generic Coefficients for All Muscle Fibers

Specific Tension: 31.8 N/cm²

Viscosity (part of FPE1): 0.01

FPE1

c1: 23.0 k1: 0.046 Lr1: 1.17

FPE2

c2: -0.02 k2: -18.7 Lr2: 0.79

FSE

cT: 27.8 kT: 0.0047 LrT: 0.964

OK Cancel Apply Help

By clicking on any of the fiber types, one can view and edit its properties.

Coefficients for "SS" Fibers

Force-Length Curve: $FL(L)$
FL_omega: FL_beta: 2.3 FL_rho: 1.62

Force-Velocity Curve: $FV(V,L)$
Vmax: -4.06 cV0: 5.88 cV1: 0.0
aV0: -4.7 aV1: 8.41 aV2: -5.31 bV: 0.18

Effective Activation: $Af(f_{eff}, L_{eff}, Y, S)$
af: 0.56 nf0: 2.11 nf1: 5.0

Activation Delay: $L_{eff}(t)$
TL: 0.088

Rise and Fall Times: $f_{eff}(t, f_{int}, L)$ & $f_{int}(t, f_{env}, L)$
Tf1: 48.4 Tf2: 32.0 Tf3: 66.4 Tf4: 35.6

Sag: $S(t, f_{eff})$
aS1: 1.0 aS2: 1.0 TS: 1.0

Yield: $Y(t)$
cY: 0.35 VY: 200.0 TY: 0.1

Energy Rate
ch0: 0.52 ch1: 0.38 ch2: 0.43 ch3: 0.16

OK Cancel Apply Help

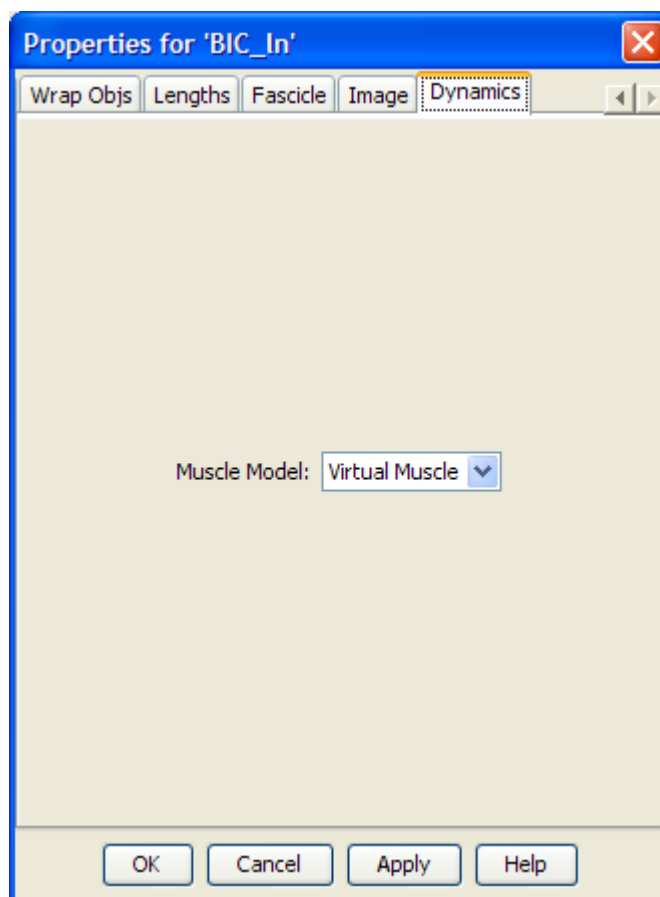
10.3.6. Image

Here, the surface color of the image can be set to blue. The transparency factor can be set to a value between 0 and 1 which corresponds to hiding and showing the image respectively.



10.3.7. Dynamics

Here, the muscle model can be set to one of: virtual model or user model.



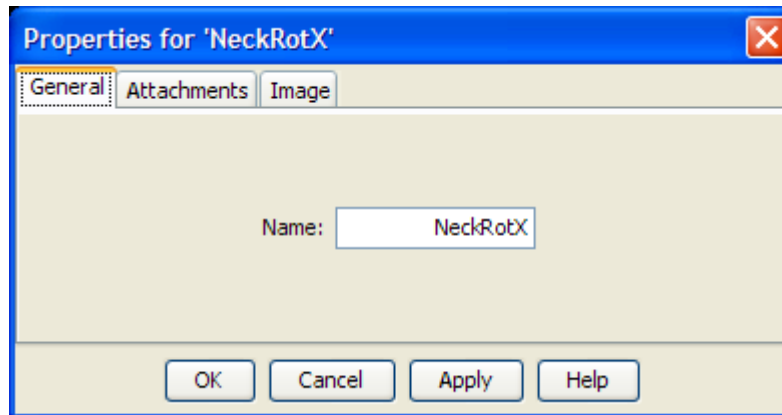
10.4. Wrapping Object

To be described.

10.5. Kinematic Driver

10.5.1. General

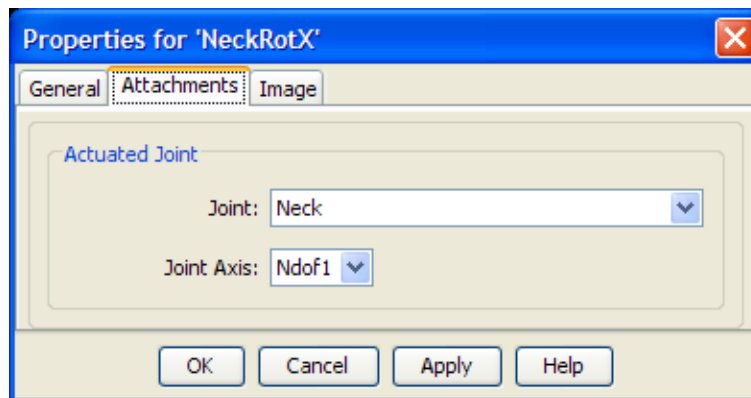
Name: A unique name for the actuator.



10.5.2. Attachments

Joint: Selects the joint to be actuated.

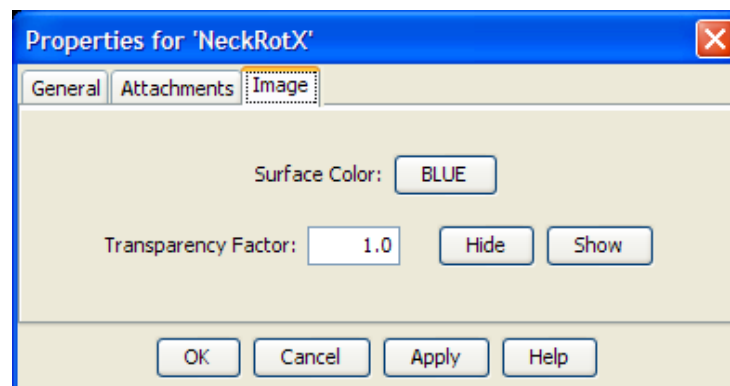
Joint Axis: Selects the joint axis to be actuated.



10.5.3. Image

Surface Color: Sets the color of the image to blue.

Transparency: Sets the transparency of the actuator image. A value of 0 shows the image, while a value of 1 hides it.

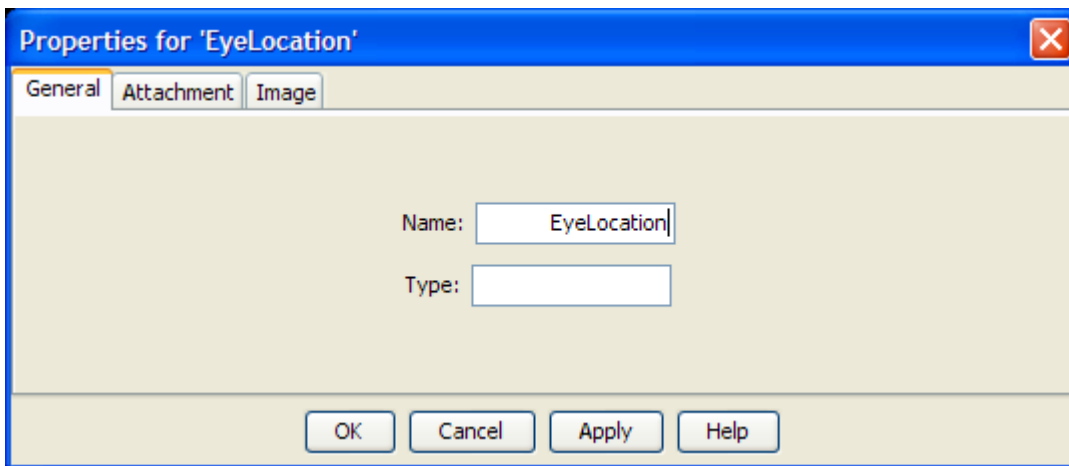


10.6. Position Sensor

10.6.1. General

Name: A unique name for the sensor.

Type: Type of the sensor.



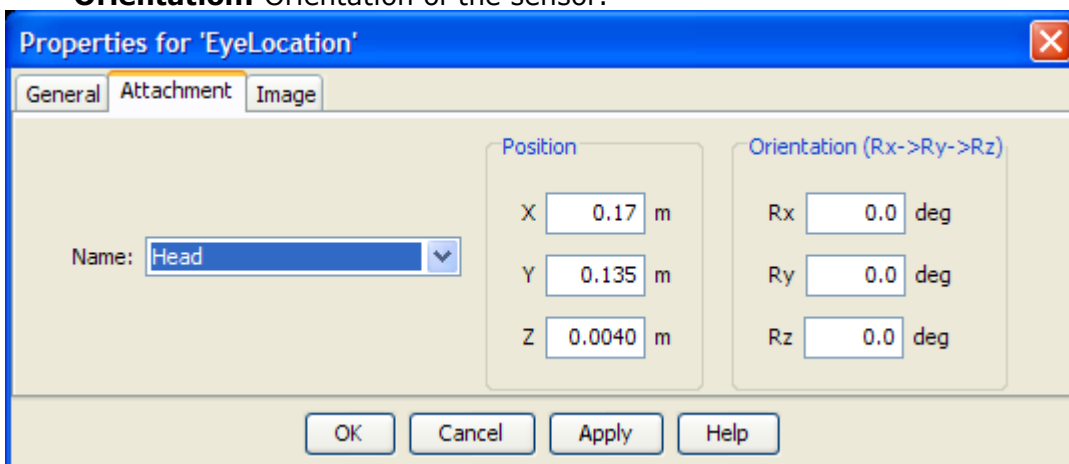
The screenshot shows a dialog box titled "Properties for 'EyeLocation'". It has three tabs: "General", "Attachment", and "Image". The "General" tab is selected. Inside the dialog, there are two input fields: "Name:" with the text "EyeLocation" and "Type:" which is empty. At the bottom, there are four buttons: "OK", "Cancel", "Apply", and "Help".

10.6.2. Attachment

Name: Selects the segment to be attached to.

Position: Position of the sensor.

Orientation: Orientation of the sensor.

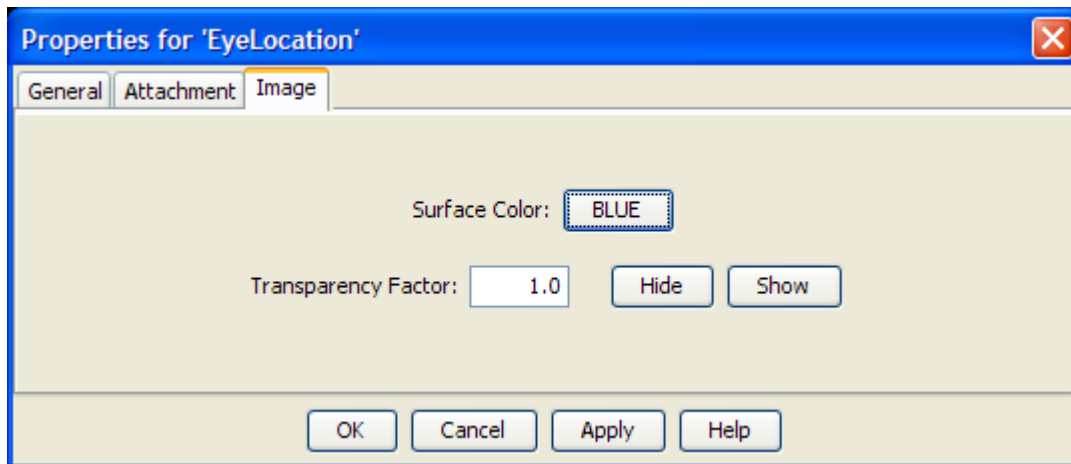


The screenshot shows the same dialog box, but with the "Attachment" tab selected. The "Name:" field is now a dropdown menu showing "Head". To the right, there are two groups of input fields. The first group, labeled "Position", contains three fields: "X" with value "0.17 m", "Y" with value "0.135 m", and "Z" with value "0.0040 m". The second group, labeled "Orientation (Rx->Ry->Rz)", contains three fields: "Rx" with value "0.0 deg", "Ry" with value "0.0 deg", and "Rz" with value "0.0 deg". The same four buttons ("OK", "Cancel", "Apply", "Help") are at the bottom.

10.6.3. Image

Surface Color: Sets the color of the image to blue.

Transparency: Sets the transparency of the actuator image. A value of 0 shows the image, while a value of 1 hides it.



10.7. Light

In the following, we give a description of the behavior of light and object colors in MSMS.

Light has only three pure colors: Red, Green, and Blue. All other light colors are combinations of these three. For example, yellow is a combination of equal amounts of red and green. A simple way to designate a color is by its RGB values. For example, bright yellow can be designated as (1.0, 1.0, 0), and dark yellow as (0.5, 0.5, 0). Likewise, white as (1.0, 1.0, 1.0), and black as (0.0, 0.0, 0.0). In MSMS's RGB color picker, the range is 0--255. Hence, white is represented as (255, 255, 255).

The color of an object is actually the color of the light that reflects from it. For example, a leaf is green because the light that is reflected from it is green. This is because the color green only reflects the green component of the light that it gets from the sun while absorbing the red and blue components. In fact, this is true for any color. The blue ocean reflects only the blue component of the 'white' light of the sun while absorbing the red and green components.

The situation is more complicated when the source light is not white, i.e., it doesn't have all three RGB colors. For example, a green object in the presence of a purely red light appears black. This is because the source light does not have any green component in it to be reflected from the object.

There are three types of light: Ambient, Diffuse, and Specular. Specular light is reflected perfectly, such as from a mirror. Diffuse reflection is imperfect, such as from a wall. Ambient is yet another type, but it is not clear what its reflection characteristics are. Note that each of these light types has a color. So in MSMS, each light type is represented by an RGB value.

There are four light sources in MSMS: Ambient, Directional, Point, and Spot. Ambient light has no position or direction. It is present everywhere, similar to

the light on a cloudy day. Directional light is emitted in one direction, but has no position, similar to the light on a sunny day. Point light has position, and is emitted in all directions, similar to a light bulb in a room. Spot light has both a position and a direction, similar to a flashlight.

The source of the light in MSMS is very important because it indicates what type of light is emitted. In MSMS, an ambient source emits ambient light; a directional source emits diffuse and specular lights; and a point source emits diffuse and specular lights.

Now let's look at the overall effect in MSMS. Let's say the color of an object is set in the GUI as:

1. Ambient: blue (0, 0, 1.0)
2. Specular: black (0, 0, 0)
3. Diffuse: black (0, 0, 0)

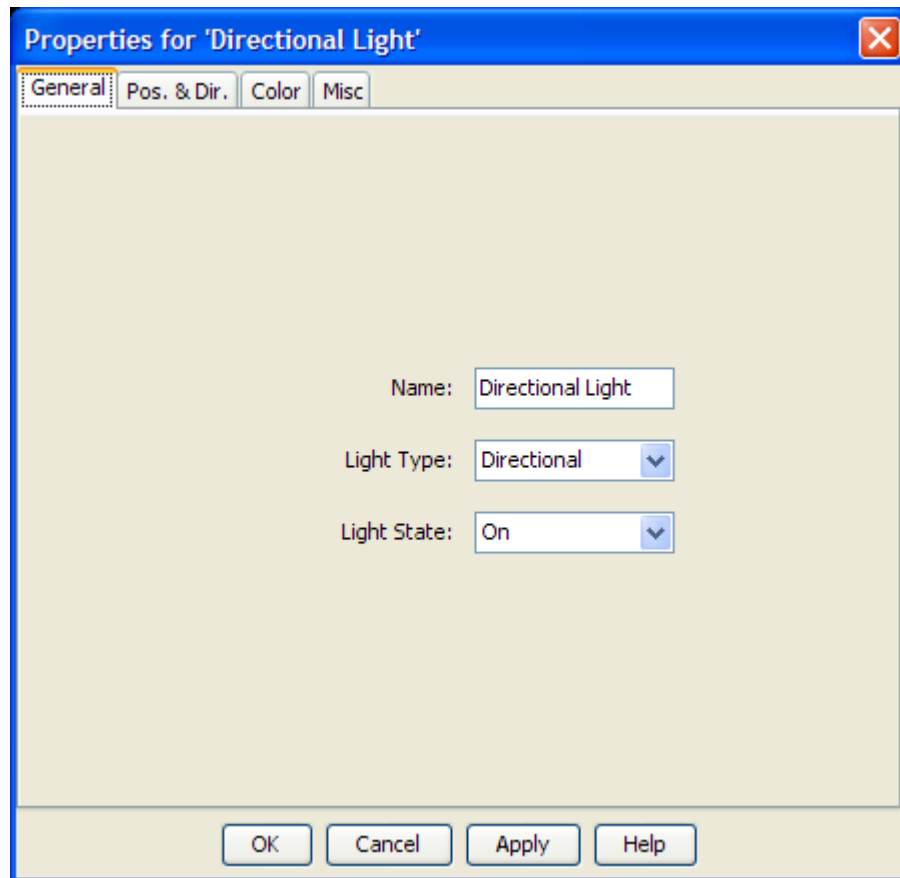
If we have a white (1.0, 1.0, 1.0) ambient source, then this source emits ambient white light. The object then reflects only the blue component of the emitted ambient light and appears blue. Now, what if the source light is changed to directional, which only emits diffuse and specular lights. Well, the object only reflects ambient light which is now non-existent. So, the object appears black.

10.7.1. General

Name: A unique name for the light source.

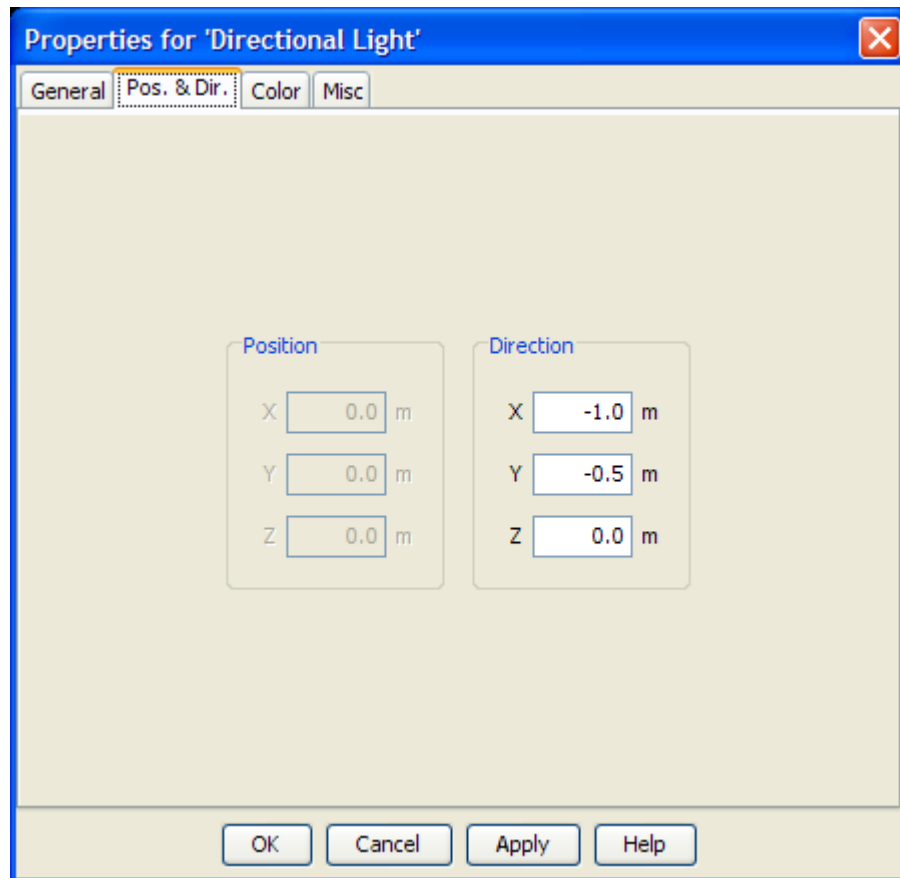
Light Type: The type of the light source can be: Ambient, Directional, Point, or Spot.

Light State: Turns the light source on and off.



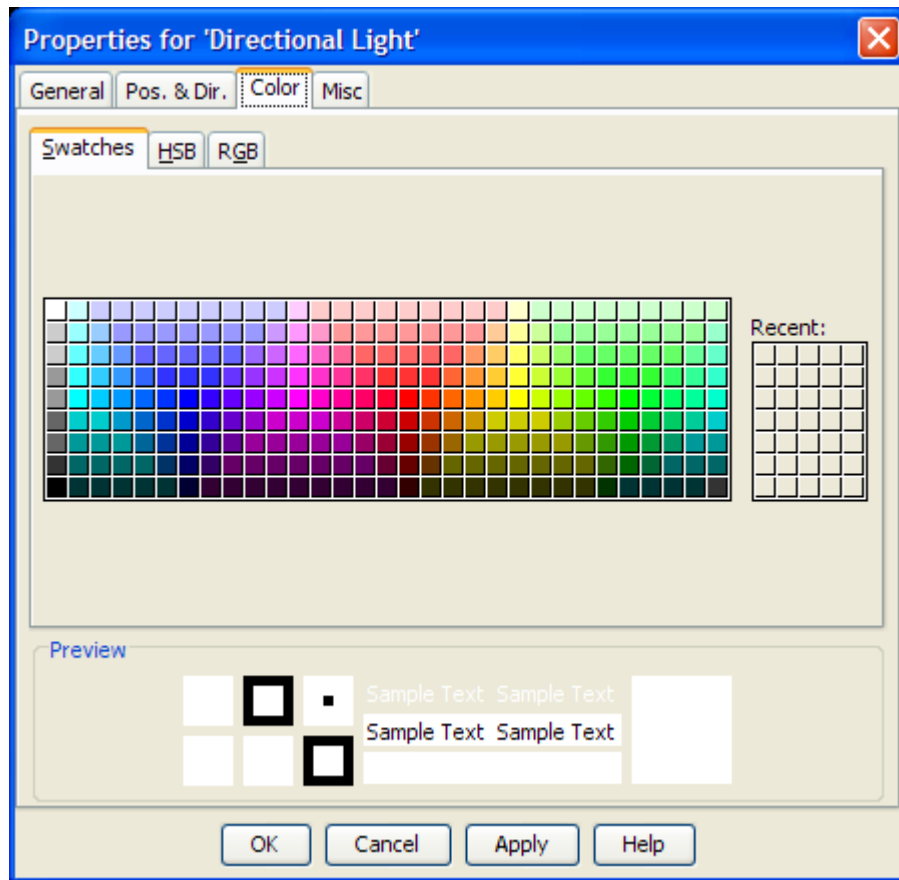
10.7.2. Position & Direction

Sets the position and direction of the light source.



10.7.3. Color

Sets the light source color.

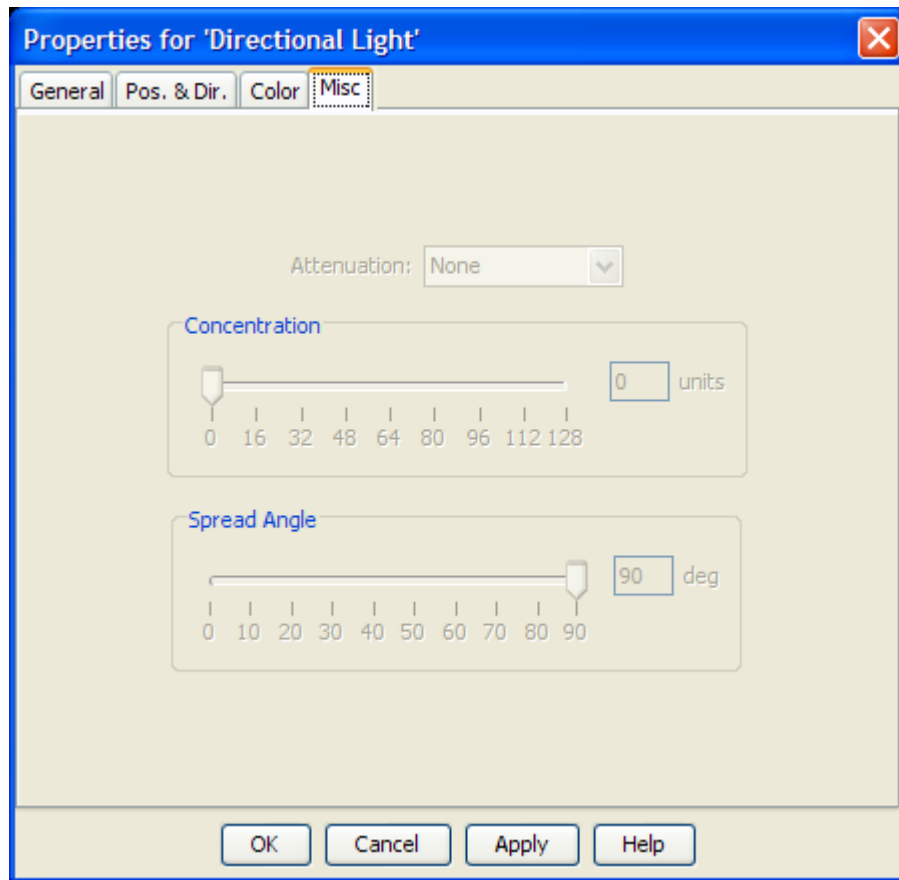


10.7.4. Miscellaneous

Attenuation: Sets the attenuation of the light source to one of: None, Linear, or Quadratic.

Concentration: Sets the concentration of the light source.

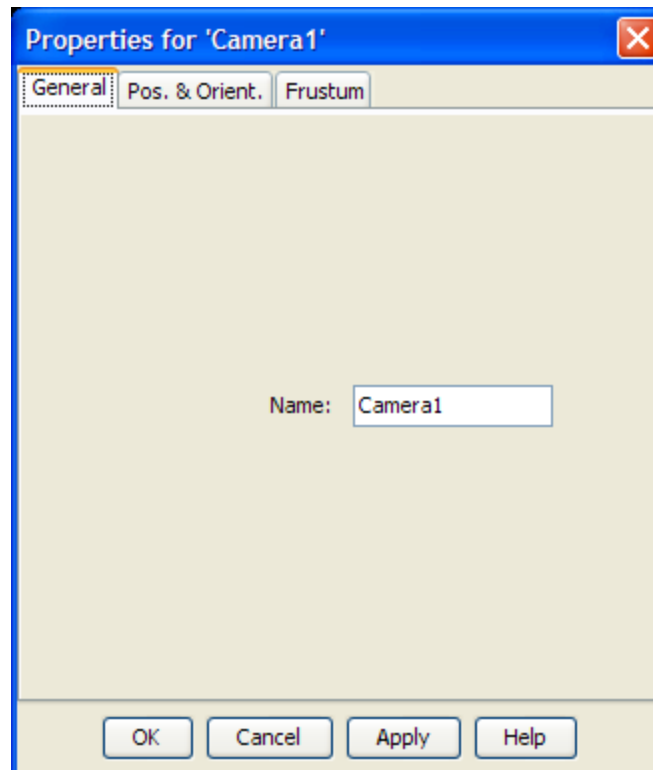
Spread Angle: Sets the spread angle of the light source.



10.8. Camera

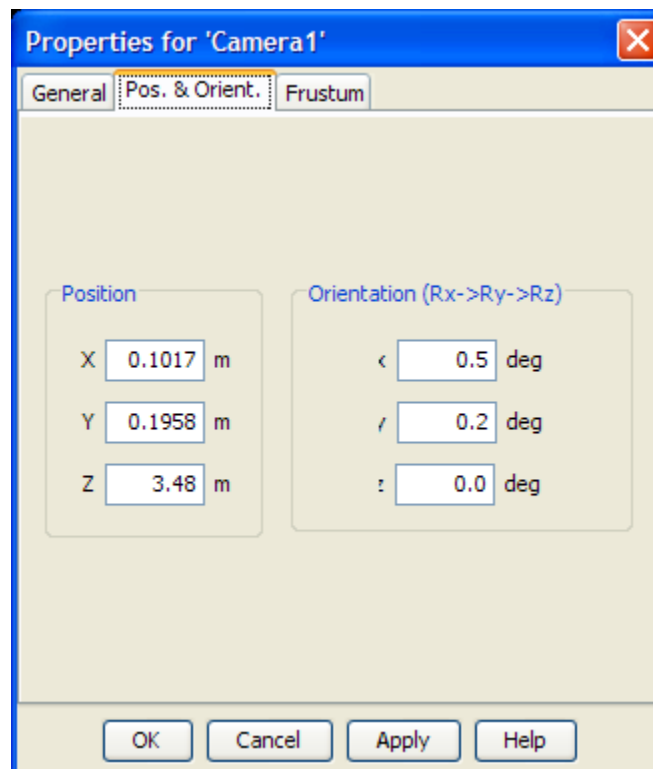
10.8.1. General

Name: Sets a unique name for the camera.



10.8.2. Position & Orientation

Sets the position and orientation of the camera.



10.8.3. Frustum

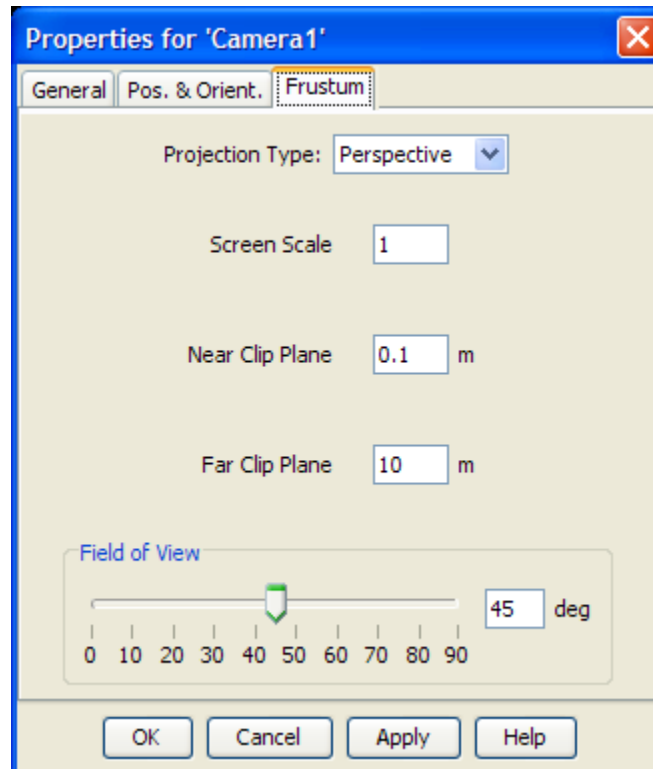
Projection Type: Sets the projection type to either Perspective or Orthographic.

Screen Scale: Sets the screen scale.

Near Clip Plane: Sets the near clip plane.

Far Clip Plane: Sets the far clip plane.

Field of View: Sets the angle of the field of view.



11. Appendix C: XML Definitions of Model Components

Components of the MSMS model must be defined in a XML file following the format specified below.

11.1. Segment

The segments components have the following structure (the tags have been explained in [\[blue\]](#)).

```
<component xsi:type="Segment" name="[segment_name]"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>
    <componentType>S</componentType>
    <componentNumber>\[a unique identifier\]</componentNumber>
  </id>
  <category>\[Human/Prosthesis/World\]</category>
  <images>
    <image>
      <name>\[unique image name\] </name>
      <scalar>\[uniform scaling factor\]</scalar>
      <xScalar>\[non-uniform scaling factor along x-axis\]</xScalar>
      <yScalar>\[non-uniform scaling factor along y-axis\]</yScalar>
      <zScalar>\[non-uniform scaling factor along z-axis\]</zScalar>
      <selected>\[select if image is displayed\]</selected>
      <collisionDetection>\[select if image is used for collision\]
      </collisionDetection>
      <shape xsi:type="Mesh"> \[More details about the shape below\]
        <xLengthM>0.0</xLengthM>
        <yLengthM>0.0</yLengthM>
        <zLengthM>0.0</zLengthM>
        <imageFilename>\[name of the image file\]</ImageFilename>
      </shape>
      <transparency>\[0-1\]</transparency>
      <offsetM>
        <x>0.0</x>
        <y>0.0</y>
        <z>0.0</z>
      </offsetM>
      <orientation>
        <r1c1>\[row 1 column 1\]</r1c1>
        <r1c2>\[row 1 column 2\]</r1c2>
        <r1c3>\[row 1 column 3\]</r1c3>
        <r2c1>\[row 2 column 1\]</r2c1>
        <r2c2>\[row 2 column 2\]</r2c2>
        <r2c3>\[row 2 column 3\]</r2c3>
        <r3c1>\[row 3 column 1\]</r3c1>
        <r3c2>\[row 3 column 2\]</r3c2>
        <r3c3>\[row 3 column 3\]</r3c3>
      </orientation>
    </image>..... \[multiple images\]
```

```

</images>
<massKg>[mass in kg of the segment]</massKg>
<massCenterM>
  <x>[center of mass x coordinate]</x>
  <y>[center of mass y coordinate]</y>
  <z>[center of mass z coordinate]</z>
</massCenterM>
<inertiaKgM2>
  <r1c1>[row 1 column 1]</r1c1>
  <r1c2>[row 1 column 2]</r1c2>
  <r1c3>[row 1 column 3]</r1c3>
  <r2c1>[row 2 column 1]</r2c1>
  <r2c2>[row 2 column 2]</r2c2>
  <r2c3>[row 2 column 3]</r2c3>
  <r3c1>[row 3 column 1]</r3c1>
  <r3c2>[row 3 column 2]</r3c2>
  <r3c3>[row 3 column 3]</r3c3>
</inertiaKgM2>
<coefficientOfFriction>[coefficient parameter]</coefficientOfFriction>
<coefficientOfRestitution>[collision parameter]</coefficientOfRestitution>
<softness>[softness parameter]</softness>
</component>

```

11.2. Joint

The joints components have the following structure (the tags have been explained inline).

```

<component xsi:type="[joint_type]" name="[joint_name]"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>
    <componentType>J</componentType>
    <componentNumber>[unique identifier]</componentNumber>
  </id>
  <parentId>
    <componentType>S</componentType>
    <componentNumber>[proximal segment id]</componentNumber>
  </parentId>
  <category>[Human/Prosthesis/World]</category>
  <outboardComponentId>
    <componentType>S</componentType>
    <componentNumber>[distal segment id]</componentNumber>
  </outboardComponentId>
  <inboardJointCenterM>
    <x>[proximal segment x-coordinate]</x>
    <y>[proximal segment y-coordinate]</y>
    <z>[proximal segment z-coordinate]</z>
  </inboardJointCenterM>
  <outboardJointCenterM>
    <x>[distal segment x-coordinate]</x>
    <y>[distal segment y-coordinate]</y>
    <z>[distal segment z-coordinate]</z>
  </outboardJointCenterM>
</component>

```

```

</outboardJointCenterM>
<initialOrientationMatrix>
  <r1c1>[row 1 column 1]</r1c1>
  <r1c2>[row 1 column 2]</r1c2>
  <r1c3>[row 1 column 3]</r1c3>
  <r2c1>[row 2 column 1]</r2c1>
  <r2c2>[row 2 column 2]</r2c2>
  <r2c3>[row 2 column 3]</r2c3>
  <r3c1>[row 3 column 1]</r3c1>
  <r3c2>[row 3 column 2]</r3c2>
  <r3c3>[row 3 column 3]</r3c3>
</initialOrientationMatrix>
<rotationAxis1>
  <dofName>[degree of freedom name]</dofName>
  <minAngleRad>[minimum angle for degree of
freedom]</minAngleRad>
  <maxAngleRad>
    [maximum angle for degree of freedom]
  </maxAngleRad>
  <defaultAngleRad>
    [default angle for degree of freedom]
  </defaultAngleRad>
  <defaultAngularVelocityRadsPerSec>
    [default angular velocity for degree of freedom]
  </defaultAngularVelocityRadsPerSec>
  <axisM>
    <x>[axis of rotation vector x co-ordinate in meters]</x>
    <y>[axis of rotation vector y co-ordinate in meters]</y>
    <z>[axis of rotation vector z co-ordinate in meters]</z>
  </rotationAxis1>
  <rotationAxis2>
    <dofName>[degree of freedom name]</dofName>
    <minAngleRad>[minimum angle for degree of
freedom]</minAngleRad>
    <maxAngleRad>
      [maximum angle for degree of freedom]
    </maxAngleRad>
    <defaultAngleRad>
      [default angle for degree of freedom]
    </defaultAngleRad>
    <defaultAngularVelocityRadsPerSec>
      [default angular velocity for degree of freedom]
    </defaultAngularVelocityRadsPerSec>
    <axisM>
      <x>[axis of rotation vector x co-ordinate in meters]</x>
      <y>[axis of rotation vector y co-ordinate in meters]</y>
      <z>[axis of rotation vector z co-ordinate in meters]</z>
    </rotationAxis2>
    <rotationAxis3>
      <dofName>[degree of freedom name]</dofName>
      <minAngleRad>[minimum angle for degree of
freedom]</minAngleRad>
      <maxAngleRad>

```

```

    [maximum angle for degree of freedom]
  </maxAngleRad>
  <defaultAngleRad>
    [default angle for degree of freedom]
  </defaultAngleRad>
  <defaultAngularVelocityRadsPerSec>
    [default angular velocity for degree of freedom]
  </defaultAngularVelocityRadsPerSec>
  <axisM>
    <x>[axis of rotation vector x co-ordinate in meters]</x>
    <y>[axis of rotation vector y co-ordinate in meters]</y>
    <z>[axis of rotation vector z co-ordinate in meters]</z>
  </rotationAxis3>
</component>

```

11.2.1. Comments:

- The id is a unique non-zero number for each component.
- The segment, joint and DOF names must be unique.
- The category identifies whether the component is part of the human, prosthesis or world model.
- The number of degrees of freedom is variable depending on the type of joint.
- The shape type can be a Mesh or a primitive (Cylinder, Box, Sphere...). For primitives, no image file is needed.

11.3. Muscle

The following is an example XML code for a muscle component with one fiber type.

```

<component xsi:type="DefaultMuscle" name="Default Muscle"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>
    <componentType>M</componentType>
    <componentNumber>17</componentNumber>
  </id>
  <category>Human</category>
  <muscleFibers>
    <muscleFiber name="SS">
      <genericCoefficients>
        <specificTension>[tension]</specificTension>
        <viscosity>[viscosity]</viscosity>

      <passiveElementForceLengtheningCoeffC1>[coefficient]</passiveElementForceLengtheningCoeffC1>

      <passiveElementForceLengtheningCoeffK1>[coefficient]</passiveElementForceLengtheningCoeffK1>

      <passiveElementForceLengtheningCoeffLr1>[coefficient]</passiveElementForceLengtheningCoeffLr1>
      <passiveElementForceShorteningCoeffC2>-
        [coefficient]</passiveElementForceShorteningCoeffC2>
    </muscleFiber>
  </muscleFibers>
</component>

```

```

    <passiveElementForceShorteningCoeffK2>-
    [coefficient]</passiveElementForceShorteningCoeffK2>

    <passiveElementForceShorteningCoeffLr2>[coefficient]</passiveElementForceShor
    teningCoeffLr2>

    <seriesElementsForceCoeffCT>[coefficient]</seriesElementsForceCoeffCT>

    <seriesElementsForceCoeffKT>[coefficient]</seriesElementsForceCoeffKT>

    <seriesElementsForceCoeffLrT>[coefficient]</seriesElementsForceCoeffLrT>
    </genericCoefficients>

    <optimalSarcomereLengthMicrom>[length]</optimalSarcomereLengthMicrom>
    <vHalf>-0.515223</vHalf>
    <fHalf>8.5</fHalf>
    <fmin>0.5</fmin>
    <fmax>2.0</fmax>
    <comment>"super-slow" soleus</comment>
    <forceLengthOmega>1.26</forceLengthOmega>
    <forceLengthBeta>2.3</forceLengthBeta>
    <forceLengthRho>1.62</forceLengthRho>
    <forceVelocityVmax>-4.06</forceVelocityVmax>
    <forceVelocityCV0>5.88</forceVelocityCV0>
    <forceVelocityCV1>0.0</forceVelocityCV1>
    <forceVelocityAV0>-4.7</forceVelocityAV0>
    <forceVelocityAV1>8.41</forceVelocityAV1>
    <forceVelocityAV2>-5.31</forceVelocityAV2>
    <forceVelocityBV>0.18</forceVelocityBV>
    <activationFrequency>0.56</activationFrequency>
    <activationNF0>2.11</activationNF0>
    <activationNF1>5.0</activationNF1>
    <activationDelay>0.088</activationDelay>
    <riseAndFallTimesTf1>48.4</riseAndFallTimesTf1>
    <riseAndFallTimesTf2>32.0</riseAndFallTimesTf2>
    <riseAndFallTimesTf3>66.4</riseAndFallTimesTf3>
    <riseAndFallTimesTf4>35.6</riseAndFallTimesTf4>
    <sagAS1>1.0</sagAS1>
    <sagAS2>1.0</sagAS2>
    <sagTS>1.0</sagTS>
    <yieldCY>0.35</yieldCY>
    <yieldTY>0.1</yieldTY>
    <yieldVY>200.0</yieldVY>
    <energyRateCh0>0.52</energyRateCh0>
    <energyRateCh1>0.38</energyRateCh1>
    <energyRateCh2>0.43</energyRateCh2>
    <energyRateCh3>0.16</energyRateCh3>
    </muscleFiber>
    <fiberRank>1</fiberRank>
    <fractionOfPCSA>0.0</fractionOfPCSA>
    <numOfMotorUnits>0</numOfMotorUnits>
    </muscleFibers>
  </component>

```

11.4. Wrapping Object

To be described.

11.5. Kinematic Driver

```
<component xsi:type="ExternalActuator"
name="ground_Shoulder_Kinematic_Driver"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>
    <componentType>EA</componentType>
    <componentNumber>[unique ID]</componentNumber>
  </id>
  <parentId>
    <componentType>J</componentType>
    <componentNumber>[unique ID]</componentNumber>
  </parentId>
  <category>Prosthesis</category>
  <jointDofId>SimMechanics_RootPart--shoulder-1</jointDofId>
  <jointDofIdAvail/>
</component>
```

11.6. Position Sensor

```
<component xsi:type="PositionSensor" name="index_Finger_position_sensor"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id>
    <componentType>PS</componentType>
    <componentNumber>[unique ID]</componentNumber>
  </id>
  <parentId>
    <componentType>S</componentType>
    <componentNumber>[unique ID]</componentNumber>
  </parentId>
  <category>Prosthesis</category>
  <positionM>
    <x>[position x coordinate]</x>
    <y>[position y coordinate]</y>
    <z>[position z coordinate]</z>
  </positionM>
  <orientationMatrix>
    <r1c1>[row 1 column 1 element]</r1c1>
    <r1c2>[row 1 column 2 element]</r1c2>
    <r1c3>[row 1 column 3 element]</r1c3>
    <r2c1>[row 2 column 1 element]</r2c1>
    <r2c2>[row 2 column 2 element]</r2c2>
    <r2c3>[row 2 column 3 element]</r2c3>
    <r3c1>[row 3 column 1 element]</r3c1>
    <r3c2>[row 3 column 2 element]</r3c2>
```

```

        <r3c3>[row 3 column 3 element]</r3c3>
    </orientationMatrix>
</component>

```

11.7. Light

There are four types of lights available in MSMS, and they are represented as follows in the `view.xml` file.

AMBIENT LIGHT

```

<light name="[light name]">
  <ns20:id xmlns:ns20="http://www.cgtechnical.com/ami/genericModel/mstns">
    <ns20:componentType>LS</ns20:componentType>
    <ns20:componentNumber>[light ID]</ns20:componentNumber>
  </ns20:id>
  <lightType>Ambient</lightType>
  <lightIsOn>[true/false]</lightIsOn>
  <color>
    <red>[1-255]</red>
    <green>[1-255]</green>
    <blue>[1-255]</blue>
  </color>
</light>

```

DIRECTIONAL LIGHT

```

<light name="[light name]">
  <ns16:id xmlns:ns16="http://www.cgtechnical.com/ami/genericModel/mstns">
    <ns16:componentType>LS</ns16:componentType>
    <ns16:componentNumber>[light ID]1</ns16:componentNumber>
  </ns16:id>
  <lightType>Directional</lightType>
  <lightIsOn>[true/false]</lightIsOn>
  <color>
    <red>[1-255]</red>
    <green>[1-255]</green>
    <blue>[1-255]</blue>
  </color>
  <direction>
    <ns17:x
xmlns:ns17="http://www.cgtechnical.com/ami/genericModel/mstns">-
1.0</ns17:x>
    <ns18:y
xmlns:ns18="http://www.cgtechnical.com/ami/genericModel/mstns">-
1.0</ns18:y>
    <ns19:z
xmlns:ns19="http://www.cgtechnical.com/ami/genericModel/mstns">1.0</ns19:z
>
  </direction>
</light>

```

POINT LIGHT


```

<light name="[light name]">
  <ns21:id xmlns:ns21="http://www.cgtechnical.com/ami/genericModel/mstns">
    <ns21:componentType>LS</ns21:componentType>
    <ns21:componentNumber>[light ID]</ns21:componentNumber>
  </ns21:id>
  <lightType>Spot</lightType>
  <lightIsOn>[true/false]</lightIsOn>
  <color>
    <red>[1-255]</red>
    <green>[1-255]</green>
    <blue>[1-255]</blue>
  </color>
  <direction>
    <ns22:x
xmlns:ns22="http://www.cgtechnical.com/ami/genericModel/mstns">0.0</ns22:x
>
    <ns23:y
xmlns:ns23="http://www.cgtechnical.com/ami/genericModel/mstns">-
1.0</ns23:y>
    <ns24:z
xmlns:ns24="http://www.cgtechnical.com/ami/genericModel/mstns">0.0</ns24:z
>
    </direction>
    <positionM>
      <ns25:x
xmlns:ns25="http://www.cgtechnical.com/ami/genericModel/mstns">-
2.0</ns25:x>
      <ns26:y
xmlns:ns26="http://www.cgtechnical.com/ami/genericModel/mstns">2.0</ns26:y
>
      <ns27:z
xmlns:ns27="http://www.cgtechnical.com/ami/genericModel/mstns">0.0</ns27:z
>
    </positionM>
    <attenuation>[none, linear, quadratic]</attenuation>
    <concentration>[0-1]</concentration>
    <spreadAngleRad>1.5707963267948966</spreadAngleRad>
</light>

```

SPOT LIGHT

```

<light name="[light name]">
  <ns21:id xmlns:ns21="http://www.cgtechnical.com/ami/genericModel/mstns">
    <ns21:componentType>LS</ns21:componentType>
    <ns21:componentNumber>[light ID]</ns21:componentNumber>
  </ns21:id>
  <lightType>Spot</lightType>
  <lightIsOn>[true/false]</lightIsOn>
  <color>
    <red>[1-255]</red>
    <green>[1-255]</green>
    <blue>[1-255]</blue>

```

```

    </color>
    <direction>
      <ns22:x
xmlns:ns22="http://www.cgtechnical.com/ami/genericModel/mstns">0.0</ns22:x
>
      <ns23:y
xmlns:ns23="http://www.cgtechnical.com/ami/genericModel/mstns">-
1.0</ns23:y>
      <ns24:z
xmlns:ns24="http://www.cgtechnical.com/ami/genericModel/mstns">0.0</ns24:z
>
    </direction>
    <positionM>
      <ns25:x
xmlns:ns25="http://www.cgtechnical.com/ami/genericModel/mstns">-
2.0</ns25:x>
      <ns26:y
xmlns:ns26="http://www.cgtechnical.com/ami/genericModel/mstns">2.0</ns26:y
>
      <ns27:z
xmlns:ns27="http://www.cgtechnical.com/ami/genericModel/mstns">0.0</ns27:z
>
    </positionM>
    <attenuation>[none, linear, quadratic]</attenuation>
    <concentration>[0-1]</concentration>
    <spreadAngleRad>1.5707963267948966</spreadAngleRad>
  </light>

```

11.8. Camera

The camera object is represented as below in the `view.xml` file.

```

<camera name="[camera name]">
  <ns1:id xmlns:ns1="http://www.cgtechnical.com/ami/genericModel/mstns">
    <ns1:componentType>C</ns1:componentType>
    <ns1:componentNumber>[light ID]</ns1:componentNumber>
  </ns1:id>
  <positionM>
    <ns2:x xmlns:ns2="http://www.cgtechnical.com/ami/genericModel/mstns">-
6.181</ns2:x>
    <ns3:y
xmlns:ns3="http://www.cgtechnical.com/ami/genericModel/mstns">1.6141</ns3:
y>
    <ns4:z
xmlns:ns4="http://www.cgtechnical.com/ami/genericModel/mstns">0.0499</ns4:
z>
  </positionM>
  <orientation>
    <ns5:r1c1
xmlns:ns5="http://www.cgtechnical.com/ami/genericModel/mstns">0.0151788450
84094327</ns5:r1c1>
    <ns6:r1c2

```

```

xmlns:ns6="http://www.cgtechnical.com/ami/genericModel/mstns">0.2469931436
5601568</ns6:r1c2>
  <ns7:r1c3
xmlns:ns7="http://www.cgtechnical.com/ami/genericModel/mstns">-
0.9688983381391925</ns7:r1c3>
  <ns8:r2c1
xmlns:ns8="http://www.cgtechnical.com/ami/genericModel/mstns">-
0.001537749961623697</ns8:r2c1>
  <ns9:r2c2
xmlns:ns9="http://www.cgtechnical.com/ami/genericModel/mstns">0.9690145937
292025</ns9:r2c2>
  <ns10:r2c3
xmlns:ns10="http://www.cgtechnical.com/ami/genericModel/mstns">0.246998689
19675574</ns10:r2c3>
  <ns11:r3c1
xmlns:ns11="http://www.cgtechnical.com/ami/genericModel/mstns">0.999883612
2204438</ns11:r3c1>
  <ns12:r3c2
xmlns:ns12="http://www.cgtechnical.com/ami/genericModel/mstns">-
0.002259231456999815</ns12:r3c2>
  <ns13:r3c3
xmlns:ns13="http://www.cgtechnical.com/ami/genericModel/mstns">0.015088336
099627064</ns13:r3c3>
</orientation>
<projectionType>\[perspective/orthographic\]</projectionType>
<screenScale>1.0</screenScale>
<fieldOfViewRad>0.7853981633974483</fieldOfViewRad>
<frontClipDistanceM>0.1</frontClipDistanceM>
<backClipDistanceM>10.0</backClipDistanceM>
</camera>

```

12. Appendix D: Motion File Formats

MSMS is capable of animating data stored in three different file formats: MSMS Motion File Format (.msm), SIMM Motion File Format (.mot), and data stored in Matlab .mat files.

12.1. MSMS Motion File Format (.msm)

MSMS motion file contains the data on the movement trajectories of the joints in a MSMS model. MSMS can read the data in the motion file and use it to animate the model. The MSMS motion file format has the extension "msm" and has to be built following the guidelines described below.

The motion file is in ASCII format and can be edited using any text editor or any program that can generate ASCII files. The motion file is composed of arbitrary number of rows. The first row is the header and contains the names of the degrees of freedom (DOF) in the model that must be animated. The following rows contain frames of data for the DOFs listed in the first row.

12.1.1. Format of the header row

Below is an example of the header row:

TIME Joint1_Tx Joint1_Ty Joint1_Tz Joint1_Rx Joint1_Ry Joint1_Rz Joint2_FE Joint2_AbAd Joint2_Rot

- The header row starts with the word "TIME", which identifies the time column, and it is followed by the names of the DOF in the animated model.
- All DOF names must be separated by spaces.
- The header is limited to a single line.
- The DOFs are either rotational or translational.
- The first element in the header is always "TIME".
- DOF names must be single strings with no space in between the characters. "Joint2_FE" is a valid name whereas "Joint2 FE" is not. The DoF names are determined by the model and the user must obtain them from the model when building the motion files.
- The header does not have to include all the DOF names. The only restriction is that if one DOF of a joint is included in the header, then all of the DOFs of the same joint **MUST** be included as well. Let's consider a hip joint that has three dofs: **hip_add**, **hip_flex**, and **hip_rot**. If, for example, **hip_flex** is included in the header, then, for the motion file to be valid, both **hip_add** and **hip_rot**, must also be included in the header.
- Each DOF in the header identifies a data column which describes the motion data for that DOF.
- The DOF names in the header do not have to be listed in any specific order. For example, the header rows below are both valid:

TIME Joint1_Tx Joint1_Ty Joint1_Tz Joint1_Rx Joint1_Ry Joint1_Rz Joint2_FE Joint2_AbAd Joint2_Rot
TIME Joint1_Ty Joint1_Tz Joint1_Tx Joint1_Rx Joint1_Rz Joint1_Ry Joint2_AbAd Joint2_Rot Joint2_FE

12.1.2. Format of the data rows

- The number of the data rows that follow the header are not limited.
- The number of the entries in each data row must match the number of the labels in the header and must follow the same order.
- Data entries in a row are separated by spaces.
- All numerical values will be stored as double-precision floating point.
- Each row of data must be on the same line (each line ends with an end-of-line (EOL) character(s), which is dependent on the operating system).
- The unit for the time data in the first column is second.
- The units of data for rotational and translational DOF are in degrees and meters, respectively. The reason for using degrees instead of the SI unit for angular position, radians, is that it is more intuitive for the users who like to inspect the motion file or edit it manually.
- No comments are allowed in the motion file.
- The MSMS motion file is compatible with and can be used by any MSMS model as long as the model has all the DOFs listed in the motion file header.

Notes:

- Spherical joints are treated differently in the motion file because their motions are represented by quaternions. To specify a spherical joint in the header, ***Mehdi, please complete this***
- The motion file does not contain data on camera position and orientation. The user can use the keyboard and mouse to position and orient the camera as desired.

12.1.3. An Example msm Motion File

| TIME | Joint1_Tx | Joint1_Ty | Joint1_Tz | Joint1_Rx | Joint1_Ry | Joint1_Rz | Joint2_FE | Joint2_AbAd | Joint2_Rot |
|-------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-------------|------------|
| 0.328 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.08625 | 0.902 | 0 |
| 0.36 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.08625 | 0.902 | 0 |
| 0.391 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.08625 | 0.902 | 0 |
| 0.438 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.08625 | 0.902 | 0 |
| 0.485 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.08625 | 0.902 | 0 |
| 0.516 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.08625 | 0.902 | 0 |
| 0.563 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.08625 | 0.902 | 0 |
| 0.61 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.08625 | 0.902 | 0 |
| 0.641 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.08625 | 0.902 | 0 |
| 0.688 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.069 | 0.902 | 0 |
| 0.735 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.069 | 0.902 | 0 |
| 0.766 | 0.03452 | 0.04704 | 0.06464 | 0.56901 | 0.21084 | -0.055849 | 0.069 | 0.902 | 0 |

12.2. SIMM Motion File Format (.mot)

The SIMM motion file format is described in SIMM's documentation and contains frames of data for all degrees of freedom of the model in rows. A sample SIMM motion file is shown below:

```

name motion_data
datacolumns 2
datarows 50
range 0.0 5.0
endheader

Shoulder_FE    Elbow_FE
0.0             0.0
0.1             0.15
0.5             0.5
...             ...

```

- datacolumns is the number of DOF in the model that are being animated.
- Datarows is the number of data frames.
- Range is the time span of animation. Sampling time can be calculated by dividing the range by the number of datarows.
- Shoulder_FE... are the names of the DOF.
- The numbers that follow are values of the DOF.

12.3. Matlab Motion File Format (.mat)

MSMS simulations in Simulink store the simulation results (joint movements) in Matlab's binary mat files. The motion data is stored in a matrix whose rows are frames of simulation data. MSMS is capable of loading these mat files and associate its data with its corresponding MSMS model and animate it.

13. Appendix E: Protocol for Live Animation Data

Using the Animation Setup, you can setup MSMS to receive and animate motion data from live sources such as a motion capture system or a simulation program. After setting up and starting the animation, MSMS will be on standby to receive animation data and use it to animate the model.

The animation data must be sent using UDP protocol and through a port number similar to that in the Animation Setup. The UDP packet must contain the animation data in a format that matches the selected animation type in the Animation Setup.

The automatically exported model to Simulink will have the necessary blocks to create the appropriate UDP packet. In this case, you don't have to worry about the format of the UDP packet. But if you are using your own program to send the animation data to MSMS, you must build your UDP packets according to the following formats.

13.1. Ordered Joint Angles

In this protocol, the positions of all joint DOF in the model must be sent in a single UDP packet to MSMS. The format of the data is single precision float and the order of the data in the UDP packet must match the order expected by MSMS. You can find the data order expected by MSMS in two ways:

1. Export the model to Simulink and then examine the Simulink block that assembles the DOF data into a single UDP packet. You will clearly see the place of each DOF in the UDP packet.
2. Examine the `simulationsetup.xml` file in the model folder. An `xml` tag in this file `<seqNum>` is used to define joint order (the order in which joint data is passed to MSMS via UDP). By examining all occurrences of this tag, you can get the order of data for all DOF in the model.

13.2. Feature Commands

One drawback of using Ordered Joint Angles protocol is that all data must be transmitted at the same time, to avoid violating the order constraints. A significant improvement on that approach is to assign an ID to each joint object, thus allowing unique specification of any joint to be commanded. This improved approach is called Feature Commands and allows you to send the data in arbitrarily small amounts. In addition, it allows you to modify parameters of other objects such as segments (e.g color and size) and also add additional elements to your VR scene such as trajectories, sounds, etc.

13.2.1. Packet Protocol for Feature Commands

The animation data is sent to MSMS via UDP communication which is a packet oriented protocol. UDP sends packets and does not keep track of them. The latest packet received by MSMS is considered to include the most recent data. Each packet includes the following basic structure:

{ [ID, F, V] [ID, F, V] [ID, F, V] }

Where,

- ID = Identifier – Identifies an object in MSMS model
- F = Feature – Indicates which feature of the object is to be modified
- V = Value – Specifies the value(s) of the specified feature

This elementary set of data is general enough to deal with motion data but also allows changing physical properties of objects such as color and size.

Data is sent using big-endian protocol, where the most-significant byte of multi-byte words is transmitted first. Note that a single packet can contain feature/value pairs for different objects (and as many pairs as will fit within the packet size constraints).

The ‘Value’ is specific to the feature it characterizes. The number of entries and their types are variable. The required values for each attribute are described below.

Identifier: The ‘Identifier’ (ID) is composed of two characters followed by a signed 16-bit integer number as defined below. The characters identify the type of component and the integer is unique value attributed to that component. The combination of both, the component type and number, uniquely identifies a component in MSMS. For example, the ID “AB32000” refers to the component AB whose number is 32000. For example, the ID can identify a segment, a joint, an actuator, a camera, etc.

Component type is a two 8-bit ASCII values specified as “Char1”+“char2”. If one character is used, it is specified as “blank” + “Char1” (e.g. “ S”, “LS”). MSMS components and their identifiers are shown in the following table.

Component number is a 16 bit signed integer (int16) and can be obtained from the MSMS model’s XML file. You can open the XML file in a text editor and find the XML description of the component by searching for its name. The component number is specified in the middle of the <componentNumber>...</componentNumber> tag. For those components that are

not explicitly specified in the XML file (e.g. Head Tracking, Sound Playback, Trajectory Path, Message Board Setup, and Message Board Write), use 0 for the component number.

| MSMS Component | Two character Identifier | ASCII Values |
|---------------------|--------------------------|--------------|
| Segment | " S" | [32][83] |
| Joint | " J" | [32][74] |
| Muscle | " M" | [32][77] |
| Light Source | "LS" | [76][83] |
| Position Sensor | "PS" | [80][83] |
| External Actuator | "EA" | [69][65] |
| | | |
| Head Tracking | "HT" | [72][84] |
| Sound File Playback | "SO" | [83][79] |
| Trajectory Path | "TR" | [84][82] |
| Message Board Setup | "BS" | [66][83] |
| Message Board Write | "BW" | [66][87] |

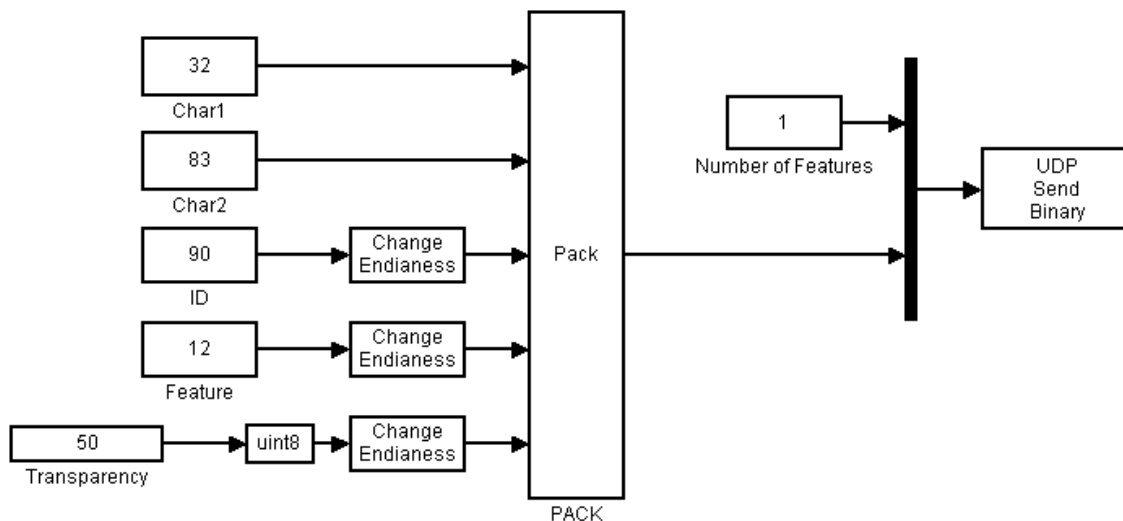
Features and Values: The Feature parameter describes what property of the component must be modified and is specified by a Feature Number provided as a 16-bit signed integer (int16). The number and the types of the data values depend on the type of feature as shown in the following table.

| Feature Type | Feature Number | Values |
|--|----------------|---|
| Motion of 1-DOF Joint | 1 | 1 single value for 1 DOF in meters and radians. Specifies the absolute translation or rotation w.r.t. the joint's zero position |
| Motion of 2-DOF Joint | 2 | 2 single values for 2 DOF in meters and radians |
| Motion of 3-DOF Joint | 3 | 3 single values for 3 DOF in meters and radians |
| Motion of 4-DOF Joint | 4 | 4 single values for 4 DOF in meters and radians |
| Motion of 5-DOF Joint | 5 | 5 single values for 5 DOF in meters and radians |
| Motion of 6-DOF Joint | 6 | 6 single values for 6 DOF in meters and radians |
| Motion of Joint as Homogeneous Transformation | 7 | 12 single values, 3 for position and 9 for 3x3 rotation matrix. $P_x, P_y, P_z, R_1C_1, R_1C_2, R_1C_3, R_2C_1, R_2C_2, R_2C_3, R_3C_1, R_3C_2, R_3C_3$ |
| Motion of Joint as 3D Position Vector | 8 | 3 single values for xyz position of the joint |
| Motion of Joint as Rotation Matrix | 9 | 9 single values for 3x3 rotation matrix of the joint orientation |
| Segment Scaling | 10 | 3 single values for scaling currently selected images of a component in X, Y, and Z axes w.r.t. the original size when the model loaded |
| Segment Color | 11 | 3 uint8 values for RGB values of the segment color. Applies to the currently visible images of a segment |
| Segment Transparency | 12 | 1 uint8 value for the transparency of the segment. Applies to currently selected images of a segment and varies between 0 (opaque) and 100 (clear) |
| Segment Visibility | 13 | 1 uint8 value for visibility of the segment (1-Visible, 0-InVisible) |
| Head Tracking | 14 | 6 single values, 3 for position and 3 for xyz orientation of the head |
| Motion of Joint with Rot. DOF as Quaternion | 15 | 4 single for the rotational DOF specified by quaternion |
| Motion of Joint with 1 Trans. DOF and Rot. DOF as quaternion | 16 | 1 single for translational DOF 4 single for the rotational DOF specified by quaternion |

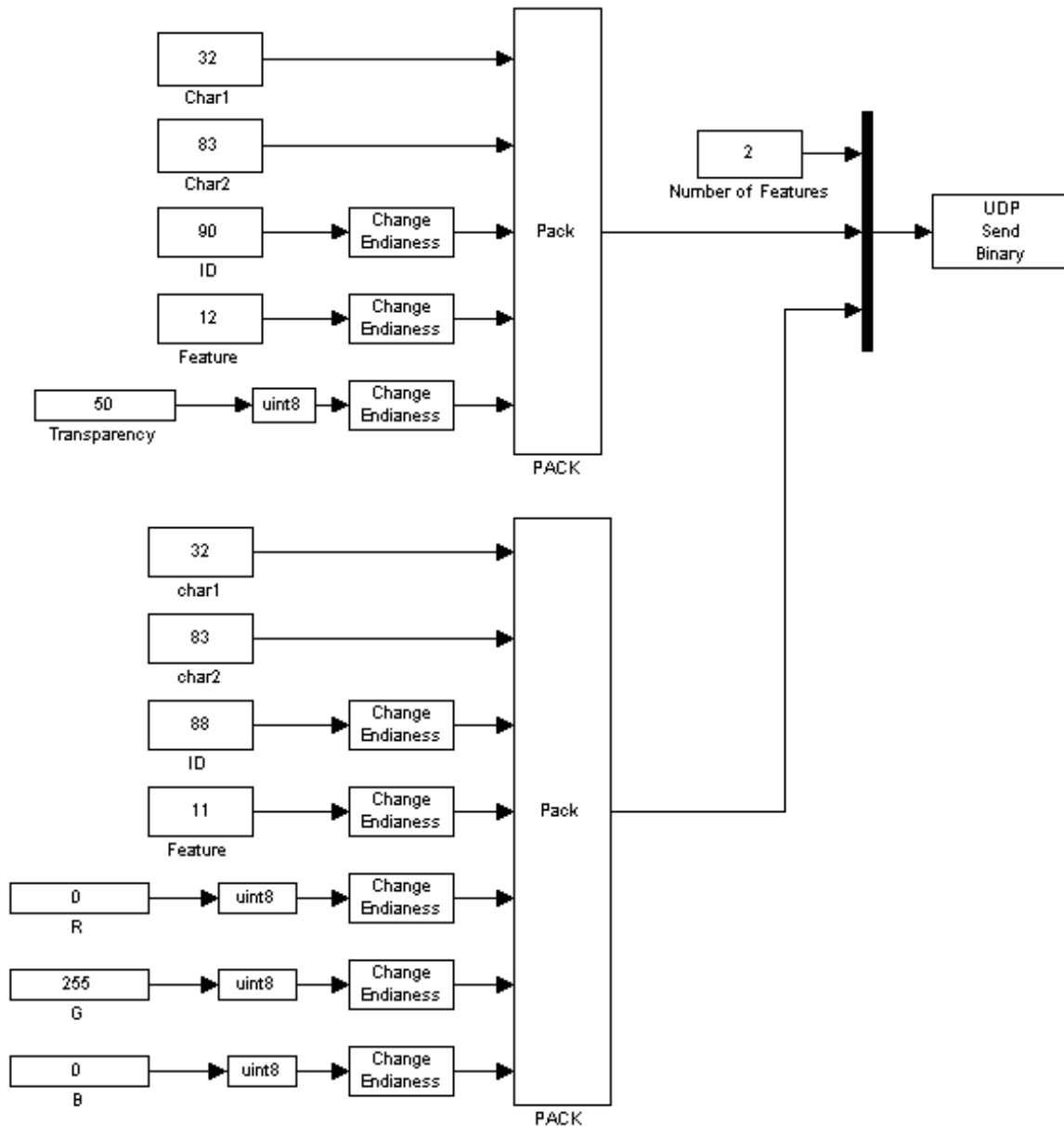
| | | |
|--|----|--|
| Motion of Joint with 2 Trans. DOF and Rot. DOF as quaternion | 17 | 2 single for translational DOF 4 single for the rotational DOF specified by quaternion |
| Motion of Joint with 3 Trans. DOF and Rot. DOF as quaternion | 18 | 3 single for translational DOF 4 single for the rotational DOF specified by quaternion |
| Trajectory Path | 19 | 1 single for diameter of the trajectory path 3 uint8 for RGB values of the trajectory color 1 uint8 for the transparency of the trajectory (0-100) 1 int16 for the number of the trajectory points 1 vector of single values specifying the xyz coordinates of the trajectory points. The length of this vector must be equal to 3 times the number of trajectory points |
| Message Board Setup | 20 | 1 uint8 to turn message board on/off (1-on, 0-off) 2 single values for x-y position of the board on the screen 3 uint8 for RGB values of the text color 1 uint8 for the background transparency (0-100) 3 uint8 for font type, style, and size 1 single for uniform scaling of the message board |
| Message Board Write | 21 | 1 int16 for the length of the text string to be written 1 string of characters (of type uint8) whose length must be equal to that specified in the first parameter |
| Sound File Playback | 22 | 1 int32 for sound file 1 unit8 for sound volume Note: You must name the sound files by a number only (e.g. 1.wav and 2.wav, etc.) and store them in a folder named Sound under model directory. The first value specifies the number in the sound file (e.g. 1, 2, etc.) that must be played by MSMS. |
| Notes: Unit8: 8-bit unsigned integer int16: 16-bit signed integer int32: 32-bit signed integer single: single precision float (32-bit IEEE 754 encoding) | | |

13.2.2. Example UDP Packets for Feature Commands

Example 1: In the following Simulink model, the Feature 12 (Transparency) of a segment whose ID is S90 is set to 50%. The Number of Features is set to 1 because only one feature in the model is modified.



Example 2: In the following Simulink model, the transparency of segment S90 is set to 50% and the color of the segment S88 is set to green (R=0, G=255, B=0). In this example, Number of features is set to 2 because two features (one segment's color and one segment's transparency) are modified.



14. Appendix F: ADL Animations

14.1. Using MSMS and PowerPoint to Animate ADL movements

This document describes the process of building a library of ADL animation files for MSMS and using Microsoft's PowerPoint to organize them into more complex ADL animation sequences. To perform ADL animations, one must create a MSMS model to visualize the movement, build a library of MSMS motion files representing simple movements, and use PowerPoint to combine the simple movements into more complex ADL animation sequences. The first two steps must be performed by expert users who are familiar with MSMS and its motion file format. The final stage, however, can be performed by non experts as long as they are familiar with the user interface of the PowerPoint application.

14.2. MSMS model

Depending on the application, models of varying complexity can be built in MSMS. For example, a model of the human hand is adequate for an application that requires animation of the grasp movement. But for animation of reaching movement, the human arm must also be modeled. The library of motion files and the ADL sequences are model dependent because they will be built using the data on the MSMS model. For example, the motion files will use the names of the joints degrees of freedom in the MSMS model to indicate what their movement is.

14.3. Library of the motion files

Once the MSMS model is built, a library of compatible motion files must be created. These files represent simple and primitive movements such as elbow flexion, palmar grasp, elbow extension, etc. and can be created following the MSMS motion file format.

These files can be created manually by entering the data in a text editor or constructed by a computer program that can write formatted data into a text file. The motion data themselves could be handcrafted by an expert, synthesized following well known bell-shaped velocity curves for human movements, or captured from subjects performing these movements using a motion capture system.

Each motion file is represented by two files: the motion file and an image representative of the movement in the motion file. The image file can be created using photography or by screen shots from MSMS screen. The motion file and the image file must have the same name but different extensions (e.g. "pinchgrip.msm" and "pinchgrasp.jpg").

The set of motion files and their corresponding image files must be stored in the model directory structure under "Model_name\data\ADL".

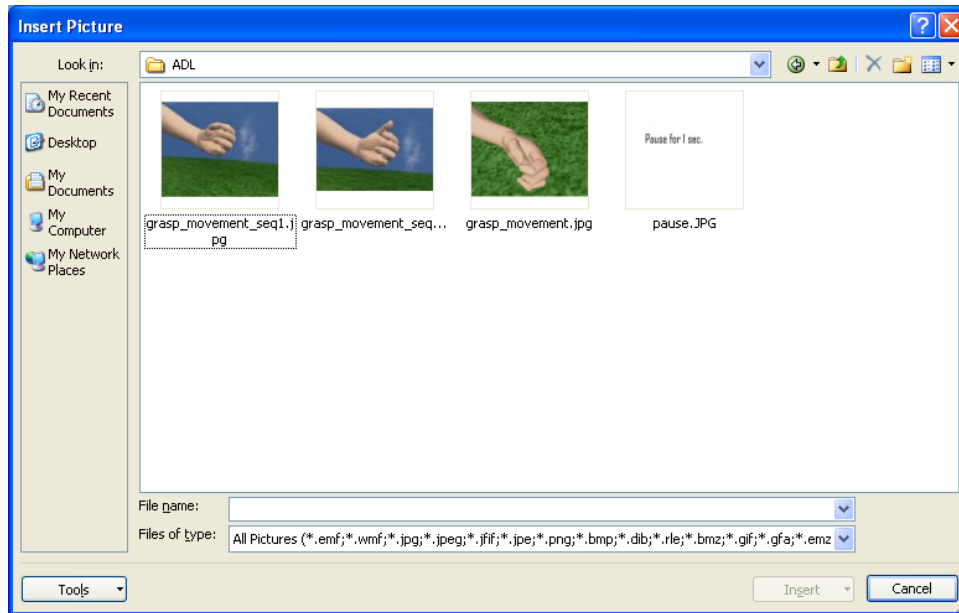
14.4. ADL Animation Sequences

The ADL animation sequences can be built using Microsoft's PowerPoint application whose interface is familiar to most clinicians. The assumption here is that a MSMS model has been

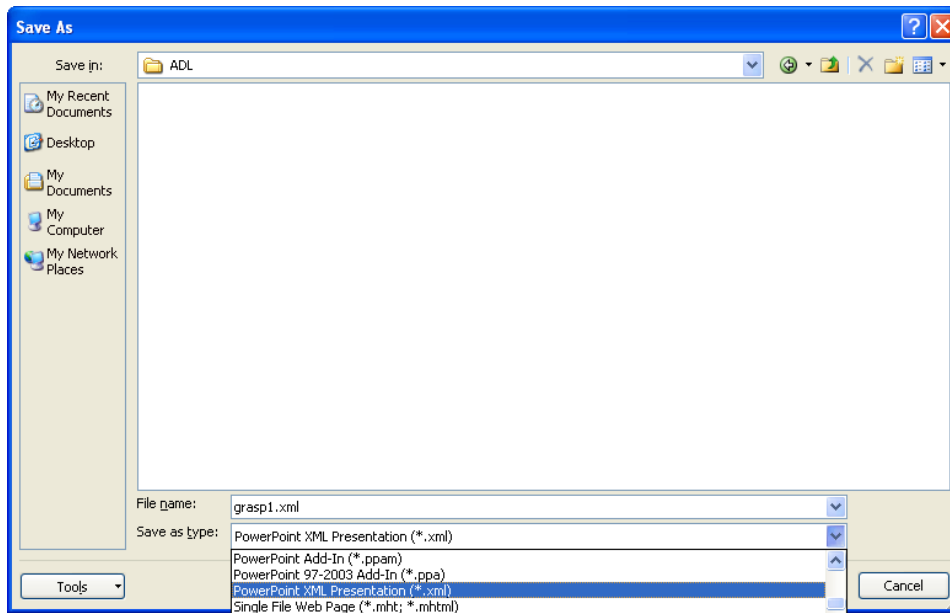
built and a library of compatible motion files have been created and stored in the models ADL folder.

The steps below describe how to create an ADL sequence in PowerPoint and animate it in MSMS. This will be followed by a more detailed set of rules for the use of PowerPoint to create ADL sequences.

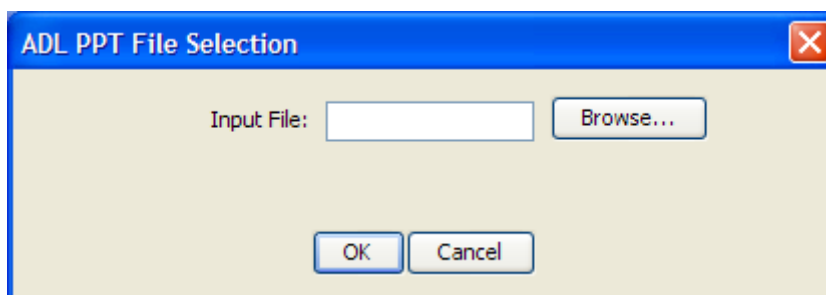
1. Open the PowerPoint and create a new presentation.
2. Insert a new slide and in the new slide insert an image corresponding to the movement you like to animate first in the ADL sequence. For this, you will need to brows to your MSMS model's ADL folder and select one of the images.



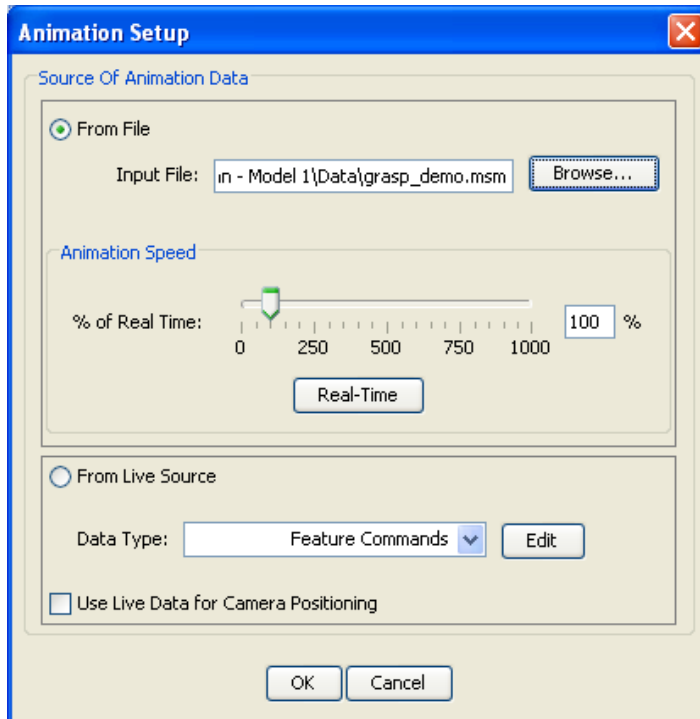
3. Repeat step (2) as many times as necessary to insert new movements into the ADL sequence.
4. Once the ADL sequence is complete, use "Save As" menu item to save the PowerPoint presentation into the model's ADL folder in **XML format** as shown in the image below (see the highlighted selection):



5. Run MSMS and open the MSMS model. For this you need to brows to the model's parent folder. Once you select the parent folder, the valid MSMS models under that folder appear in the "Simulation Configurations" list. If it doesn't, you are in the wrong folder and need to go back and select the model's parent folder. Double-click on the model name to open.
6. Once the model is loaded, Click on "Animation>ADL File Parsing" menu item. In the popup window brows to the model's ADL folder and select the XML file created in step (5) above and Click OK. This creates a MSMS motion file for the whole ADL sequence and stores it in the model's data folder, "Model_name\data". The created motion file for the ADL sequence has the same name as the PowerPoint presentation file but its extension is "msm".



7. To animate the ADL sequence created in step (6), click on "Animation> Setup..."
8. In the Animation setup window, select "From File" radio button. Then brows to the model's data folder, select the motion file for the ADL sequence, and click "OK" to close the "Animation Setup" window.



9. The play icon in the MSMS toolbar must now turn to green. Click on it to animate the ADL sequence.

14.5. Rules and Guidelines for Creation of ADL sequence in PowerPoint

- **Repetition of the same movement** – To repeat a movement more than once, duplicate the corresponding slide as many times as necessary.
- **Execution time** – To change the duration of the animation specified in a slide, enter the "Animations" tab in PowerPoint and enter the duration time into the field labeled "Automatically after". The number here will determine the length of time it will take to complete the animation in the slide. The time entered is of the form "00:00.00", indicating "minutes:seconds.hundredths of seconds"
- **Delays/Pauses between movements** – To insert a delay or pause between two animations, insert a blank slide between their corresponding slides. Then insert an image into the blank slide whose name is pause (e.g. pause.jpg). The default delay/pause time is 1 second. If you want to change the duration of the pause, enter it into the "Automatically after" field of the "Animations" tab.

15. Appendix G: Special Features

This appendix describes the features that have been developed to specifically address the needs of a specific MSMS user or application. These features therefore are application specific and are not generally useful. But we are describing them here because there may be users with similar applications and issues who might find them useful.

15.1. Blanking the model screen

One of the MSMS users, who employ MSMS to display virtual objects to the monkeys in cortical control experiments, needed the ability to make the MSMS model disappear by clicking on the spacebar in the keyboard. This feature is implemented in MSMS and can be enabled by the interested users by adding the following line to msms.properties file:

```
kpaulEnabled
```

msms.properties file is located in MSMS root directory and can be edited by any text editor such as Notepad. The presence of this line will direct MSMS to enable the blanking feature.

To use this feature, open a model and repeatedly press on the spacebar to switch between the model and the blank windows.
For this to work, the focus must be on the model area, which is done by clicking once in the model area.

15.2. Writing animation events to file

The same MSMS user, who employed MSMS to display virtual objects to the monkeys in cortical control experiments, also needed the ability to record the sequence of the following MSMS animation events in a text file.

- 1 - Animation packet received
- 2 - Rendering started
- 3 - Rendering completed
- 4 - Ready to receive packet

This feature is implemented and if it is enabled, it will write the occurrence of the above events (represented by their respective numbers) into a text file.

To enable this feature, Create an empty text file (e.g. myevents.txt) and save it in your desired directory (e.g. C:\mydir). Note that MSMS will not create myevents.txt. It has to be created first by the user in the specified directory. Then add the following two lines to msms.properties (if not already present):

```
kpaulEnabled  
kpaulPipeFilename = C:\\mydir\\myevent.txt
```

Load your model to MSMS and animate it by feature command animation. After the animation, you can see the sequence of animation events handled by MSMS in myevents.txt.

15.3. Ordered Joint Angles – APL

In animation setup window, there are three choices for animation data type from live sources. Only the first two (Feature Commands and Ordered Joint Angles) must be used by general MSMS users. These are described above in "Animation Menu\Setup" section. The third choice, "Ordered Joint Angles - APL", was developed specifically for one of the MSMS users who had written their own code to append additional animation data for changing object properties such as its color to the end of the joint angles data and send it to MSMS. The development of the Feature Command animation has made this option obsolete because it allows the Feature Command animation allows the users to animate the joints and change object properties using a well defined protocol. The third animation data type however, is kept in MSMS as a legacy code, to ensure compatibility with the older simulation programs developed by this specific user.