Specification
of
MLCad specific extensions to the Ldraw commands

1. CONTENTS

2. LIZENCE AGREEMENT

The commands described in this document are special extensions to the Ldraw command syntax. Other developers may include the same functionality in their own developments to be compatible to MLCad.

When implementing one or more of these commands the developer must clearly state the implemented commands, in a documentation available to the end user. Also a reference to this document which can be found on the following web site (http://www.lm-software.com/mlcad) has to appear in that documentation.

This specification document must not be provided to third parties on any other location than the web site mentioned above neither electronically nor on paper, without written acceptance from the author of this document.

3. MLCAD LANGUAGE EXTENSIONS

3.1 View Rotation Commands

The rotation view commands are partially taken over from LDView and have been extended to allow more flexibility. Generally they are an extension to the plain STEP command of Ldraw. Drawing stops after executing the command. If you want to make the view rotation visible a plain STEP command should be used immediately before the rotation step command.

There are four different rotation step commands:

3.1.1. Relative Rotation Steps

Relative rotation steps are based on the actual angles of the individual viewing areas. The model will rotated by the specified rotation angles relative to the current view angle.
That means if a view shows the model from it's front side, and the rotation step will rotate the model by 90° clockwise then after executing the step command you can see the model from the right side. If the current view angle is top then after executing the command, you will still see the model from top, but rotated 90° clockwise.

The syntax for this command is:

0 ROTSTEP <x-angle> <y-angle> <z-angle> [REL]

The keyword REL is optional but should be specified for clearness.

x-angle, y-angle and z-angle are the individual rotation angles for the different axes in degree (-360 to 360).

3.1.2. Additative Rotation Steps

This kind of rotation step turns the model by the specified angles, taking the current view angle into account. The command can be used to continuously rotated the model by specific angle. For example if this command is executed four times on a front view, and the model is rotated 90° clockwise on the y-axle then you will see the model from each of the four sides.

The syntax for this command is:

0 ROTSTEP <x-angle> <y-angle> <z-angle> ADD

x-angle, y-angle and z-angle are the individual rotation angles for the different axes in degree (-360 to 360).

3.1.3. Absolute Rotation Steps

The last type of rotation steps is similar to relative rotation steps, but it ignores the current view angles so that after executing this command each view area will show the same image.

The syntax for this command is:

0 ROTSTEP <x-angle> <y-angle> <z-angle> ABS

x-angle, y-angle and z-angle are the individual rotation angles for the different axes in degree (-360 to 360).

3.1.4. Rotation End Step

This command returns to the original viewing angles of the individual views.

The syntax for this command is:

0 ROTSTEP END

3.1.5. Matrix Calculation Of Rotation Steps

For each rotation step command (except END) the rotation matrix is computed by the following formula:

wx, wy and wz are the angles in radiants

s1 = sin(wx) s2 = sin(wy) s3 = sin(wz)
c1 = cos(wx) c2 = cos(wy) c3 = cos(wz)

a = c2 * c3
b = -c2 * s3
c = s2
d = c1 * s3 + s1 * s2 * c3
e = c1 * c3 - s1 * s2 * s3
f = -s1 * c2
g = s1 * s3 - c1 * s2 * c3
h = s1 * c3 + c1 * s2 * s3
i = c1 * c2

The resulting view-angle to be used by each individual view is computed in the following way, where "newmatrix" is the new matrix to be used by the view, <rotation matrix> is the matrix computed for the command. Multiplication is the standard way for multiplying matrixes.

For relative rotation steps the new view matrix is:
newmatrix = <view default matrix> * <rotation matrix>

For additative rotation steps the new view matrix is:
newmatrix = <current view matrix> * <rotation matrix>

For absolute rotation steps the new view matrix is replaced with the rotation matrix:
newmatrix = <rotation matrix>

3.2 Background Command

The background command is used to display a picture in the background of the view. The picture itself will be always behind the model, so that the model is fully visible to the viewer.

MLCad displays the image after executing the command. Another background command replaces previous background commands.

The syntax for this command is:

0 BACKGROUND <filename>

<filename> is the name of the image file. The filename can include a path name. It is recommended to set the file name in quotations marks (“). Using quotation marks also allows the use of blanks in the filename and thus avoids problems with opening the file.

3.3 Buffer Exchange Command

The buffer exchange command allows the user to create animation like instructions. It more a trick to get building instructions like the real once.

The buffer command either stores an image or retrieves a stored image. The image is taken from the actual view and when restored put in there. Es an example imagine you want to display where a motor has to be put on the chassis of a car. So you first create a building step with the motor exposed, possibly with an arrow to the point where the motor should be placed and in the next picture the motor should be there where it is supposed to be. This effect can be done as follows:
You first create the model completely without the motor. Then you save an image of the current view, and after that you put the motor above the model. When the user continues drawing you retrieve the previously saved image and redraw the motor in it’s final position.

This command is ignored if the model containing this commands is used as a sub-model.

With the help of the buffer command you can save up to eight independent images

The syntax of this command is:

0 BUFEXCHG <A-H> STORE | RETRIEVE

<A-H> the letter identifier for the storage to use
STORE .. to save the image or
RETRIVE .. to load the image back to the view

3.4 Ghost Commands

Ghost commands are a special form of other commands. Any MLCad or Ldraw command can be a ghost command. To make a command to a ghost command simply put “0 GHOST” at the beginning of the line.

So what’s the deal with this type of command?
MLCad displays ghost commands only if the model containing the command is the main model. If the model is a sub-model of another or simply a part included into a model the ghost lines are simply ignored. They are useful in connection with the buffer exchange command and allow to show detailed assembly instructions of a part when it’s viewed by its own, but if it’s included as a part these steps will not be shown.
Without this ghost command you would see things between buffer exchange commands uncontrolled since buffer exchange commands are also ignored in sub-models.
You will find some examples for the use of this command on http://www.lm-software.com/mlcad

The general syntax of this command is:

0 GHOST <other ldraw or mlcad command>

3.5 Group Commands

For simpler editing purposes MLCad allows the definition of groups, where two ore more parts are put together in a single virtual part. You can nearly everything with such a group as you can do with a normal part, except single part modifications are not possible. To keep the information about grouping after reloading a saved model, MLCad uses the group command to identify a group.

The syntax of this command is:

0 GROUP <n> <name>

where <n> is the number of commands belonging to this group and <name> is the name for this group.

The commands belonging to the group are identified by the MLCAD BTG command.

You must not use recursive groups!

3.6 Group Mark Commands

Group mark commands are used to identify that a part belongs to a group. The line following this command is selected to be included into the group identified by the BTG command.

Syntax:
0 MLCAD BTG <group name>
x ….

<group name> is the name of the group to which the following line belongs to.

The line following this statement is put into the specified named group.

3.7 Skip Section Commands

For future use MLCad already includes the functionality to skip a certain section of the file in a controlled manner. Therefore two commands are defined which identify the begin and the end of the section to skip:

0 MLCAD SKIP_BEGIN
…
0 MLCAD SKIP_END

Everything between the two lines including the SKIP commands themselves are not loaded into the memory when reading the file.

In the future several commands will be added which will be processed in MLCad in a special way. When the model is then saved replacement commands for none MLCad compatible viewers are written inside the skip section.