A Beginner's Guide to BRL-CAD

The main aim of the document - Introduction to BRL-CAD
This document aims at introducing you to BRL-CAD through a basic step-by-step tutorial. In this tutorial, you will model a chess set in BRL-CAD and by the end, we hope this tutorial changes the way you look at shapes. Brace yourself, you are about to enter the amazing world of 3D computer-aided design (CAD) modeling.

1. **What is modeling**
   A model of an object can be used to understand how it works, to visualize, analyze, print or simply used to study the object. Now, CAD modeling is the act of creating this computer representation with specific dimensions. BRL-CAD is used to model 3D objects that have all the physical characteristics of an object.

2. **Intro to BRL-CAD**
   BRL-CAD is a powerful 3D solid modeling system. This software has been in use for a wide range of military, educational, and industrial applications. CAD software like BRL-CAD requires a lot of practice to master, but this tutorial will help you get started. As you work through the tutorial, you will see commands taking shape into something you want to model.

   To get to know more about BRL-CAD, its origin, history, you can check out the following link:
   BRL-CAD FAQs

   Let's begin this journey and get BRL-CAD installed in your system. Downloading BRL-CAD is just a few clicks:
   - Go to https://brlcad.org/ and click on the **Download** link at the top left corner.
   - On the next page, click on the folder for your computer. For Windows or Mac operating systems, select **BRL-CAD for Windows** or **BRL-CAD for Mac OS X** respectively.

- Download and run the installer. Follow the installation prompts.

For a more detailed installation walkthrough, see this tutorial for newbies.

3. Intro to MGED

BRL-CAD has many applications, but in this tutorial you will work in an editor called "MGED". The Multi-Device Geometry Editor (MGED) is the main way to create geometry in BRL-CAD.

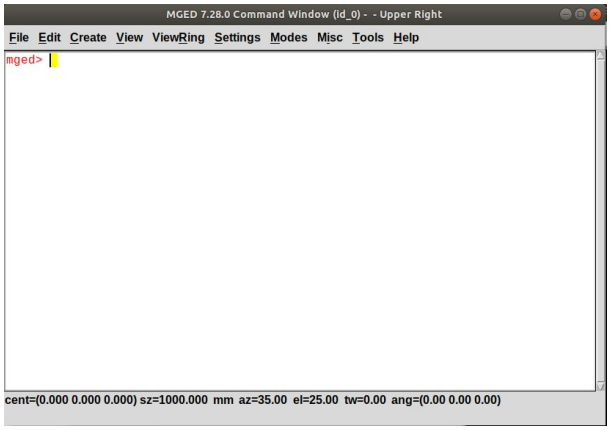
Launching MGED on Windows or Mac OS X:

- Double-click the application icon for MGED.

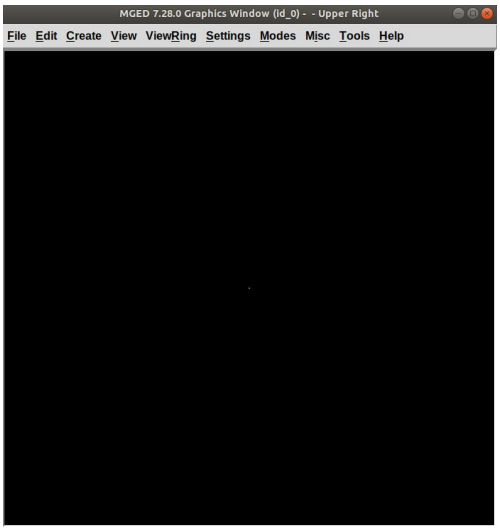
Launching MGED on Linux:

- Open a terminal (tty) prompt, using Ctrl+Alt+T (if you are on Ubuntu). Type **`mged`** in the terminal prompt and then press **`ENTER`**.
- There is a chance that you might get an error when you type **`mged`** in the command prompt stating **`mged is not recognized as a command`** or **`mged command not found`**
  In that case, you need to specify the full path to where you installed BRL-CAD.

Two new windows should pop up: the MGED Command Window and the MGED Graphics Window (sometimes called the Geometry Window). The former is for entering commands and later for displaying the output.



MGED Command Window



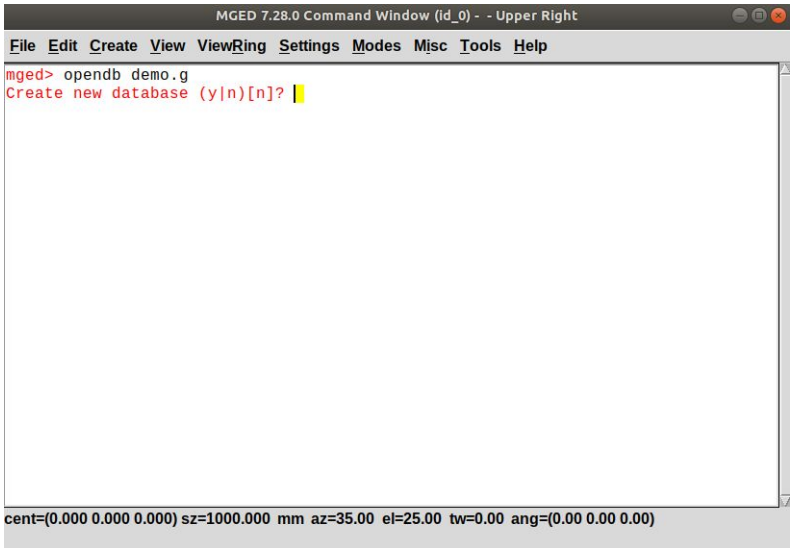
MGED Graphics Window

Both windows are empty at the moment.

While there are ways to do nearly everything in MGED via the graphical user interface menus, this tutorial is going to introduce you to modeling by way of MGED commands.

4. **Opening a database:**
   Before you start modeling anything in MGED, create a database. Move your mouse over or click anywhere on the Command Window to make it active and type **opendb** followed by the name of the database with the **.g** extension:
   opendb demo.g<ENTER>

   If the database with name **demo.g** doesn't already exist or you are creating a new one, the Command Window confirms if you want to create a new database as shown below:



Type **y** and press **ENTER**.

If **demo.g** already exists, the database opens up.

5. Modeling in BRL-CAD

Let's understand in brief how modeling in BRL-CAD works:

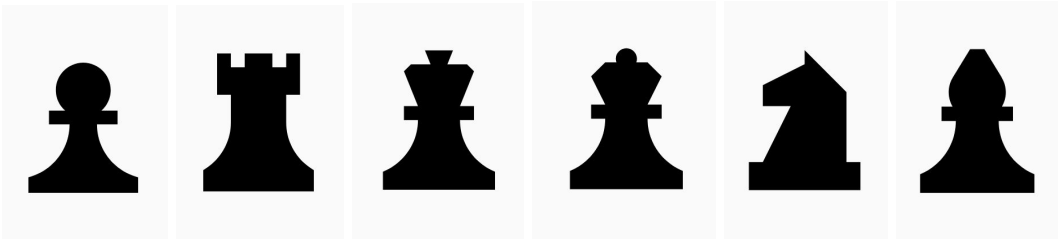
- **Using primitive shapes**: Now, what are primitive shapes? A primitive shape is a basic 3D object whose parameters can be changed without changing the shape's type. For example, a sphere is a basic 3D shape and is one of over two dozen primitives available in BRL-CAD. Click here to see the primitives used in BRL-CAD.
- **Employing basic Boolean operations on the shapes**: Not every shape we model is a primitive shape; so, to get the required output we apply basic boolean operations of union, subtraction, and intersection on these shapes. For example, a hollow cylinder can be made by subtracting a cylinder from another larger cylinder.

Once you are comfortable with a few of the commands, it will get easier to model anything to everything.

6. The modeling target:

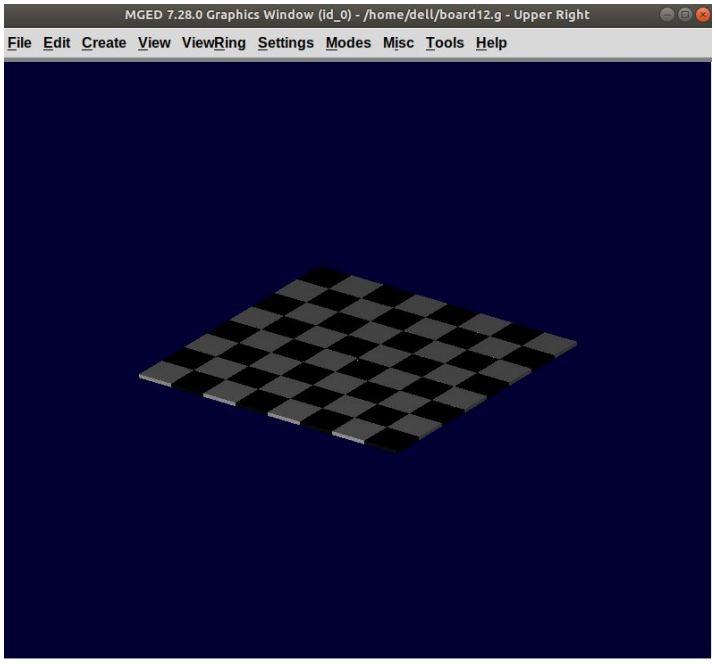
In this tutorial, you are going to model a complete chess set. We're going to base our 3D modeling on a 2D design by Arthur Shlain.

Seeing your chess pieces and chess board taking shape with each command is going to be thrilling. First, you will model all the unique chess pieces which include a pawn, rook, king, queen, knight, and bishop:



Click here to see other chess piece designs by Arthur Shlain.

After modeling the chess pieces, you will move on to the chessboard which will look like this:



7. Modeling Pawn

Let's begin by creating a new database for your pawn and name it **pawn.g**. As discussed earlier, to create a new database type the following command in the Command WIndow:

$ opendb pawn.g<ENTER>

Creating a cylinder for the base

Begin by making the Command Window active (usually by clicking anywhere in the window). To make the right circular cylinder, type the following command in the MGED prompt

in base.rcc rcc <ENTER>

Here, **in** is the command which is used to insert a primitive shape, **base.rcc** is the name of the shape and **rcc** means it is a right circular cylinder.
MGED asks you to enter **x**, **y** and **z** values of the vertex (where you want to place the center of the bottom of the shape). Type:

0 0 0 <ENTER>

Make sure to add spaces between the values.

Next, MGED will ask you to enter the x, y and z values of the height

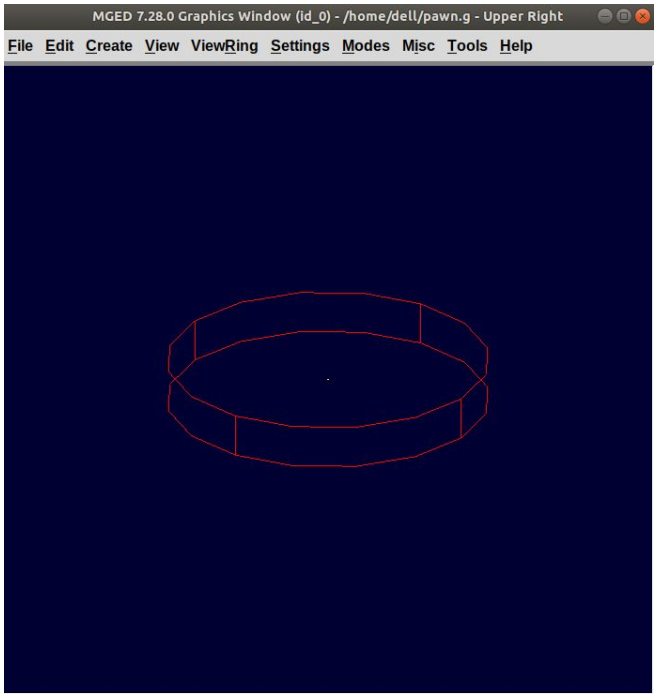
(H) vector (the height of your cylinder). Type:
0 0 0.6 <ENTER>

Then, the last value MGED will ask you to enter is the radius. Type:
2.25 <ENTER>

Your MGED command window will look something like:

```
mged>in base.rcc rc
Enter X, Y, Z of vertex: 0 0 0
Enter X, Y, Z of height (H) vector: 0 0 0.6
Enter radius: 2.25
Base.rcc
```

You will get something like this on your Graphics Window:



Rather than following this lengthy method, there is another short way to use the **in** command. It allows entering all the parameters in one go. The above command can also be written as:
in base.rcc rcc 0 0 0 0 0 0.6 2.25<ENTER>

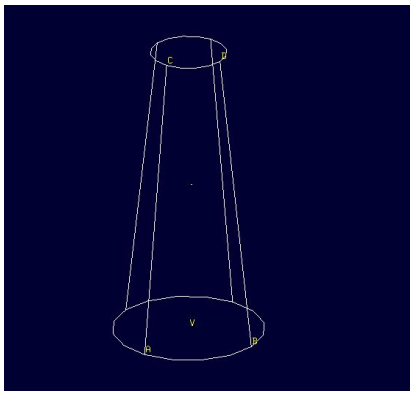
Meaning of the above command is:
in: Insert a primitive shape

base.rcc: Name it base.rcc
rcc: Shape should be a right circular cylinder
0: x value of the vertex is 0
0: y value of the vertex is 0
0: z value of the vertex is 0
0: x value of the height vector is 0
0: y value of the height vector is 0
0.6: z value of the height vector is 0.6
2.25: radius is 2.25

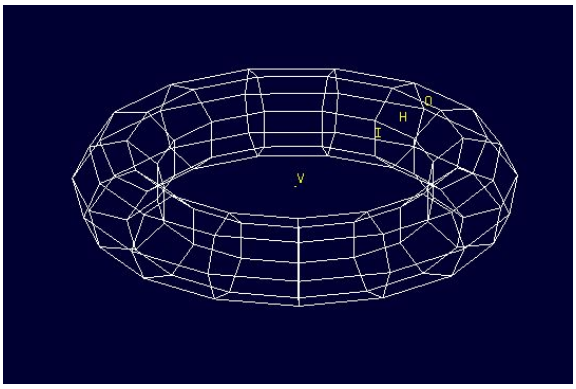
This is how you will be making the rest of the shapes. Moving on to the upper portion of the pawn.

Making the curve

This portion is a little tricky. To make the curve, you will first make a Truncated Right Cone (trc) and then subtract a Torus(tor) from the outer portion of trc.



trc



tor

To make the trc, type:

in body.trc trc<ENTER>

The **trc** should start from the top of the **rcc** i.e., at the height of **0.6**. MGED will ask for the **x**, **y**, **z** values of the vertex (center of the bottom part). Type:

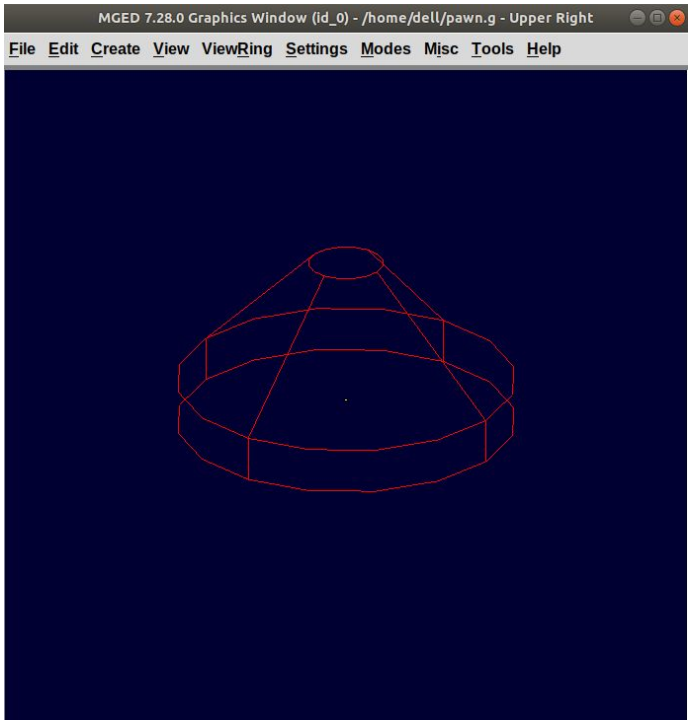
0 0 0.6<ENTER>

Then MGED will ask us to enter **x**, **y**, z values of height vector. Type:

0 0 1.7<ENTER>

The next entry we have to make is the radius of the base which must be the same as the radius of the **base.rcc**. Therefore, type:
2.25<ENTER>

The last value MGED asks for is the top radius. Type:
0.5<ENTER>
The graphics window will look like:

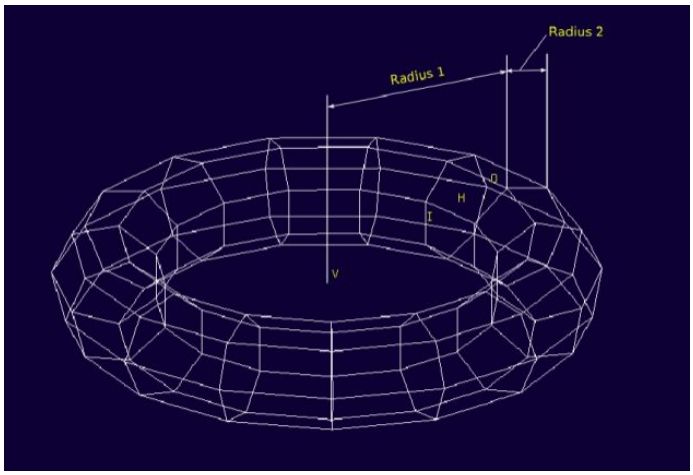


To make the curve use the short-hand method of using the **in** command. Type in the Command window:
in curve.tor tor 0 0 2.8 0 0 1 2.85 2.35<ENTER>

Here,
0 0 2.8 are the **x**, **y**, **z** values of the vertex where **2.8** = **0.6** (z value of vertex of **body.trc**) + **1.7** (height of **body.trc**) +**0.5** (radius of the top of **body.trc**).
0 0 1 are the x, y, z values of the normal vector to make the tube perpendicular to the z-axis.
2.85 is radius 1 (radius from Vertex to the center of the tube).

2.35 is radius 2 (radius of the tube).

The following image visually explains radius 1 and radius 2.



Making a cylinder for the neck

The cylinder should have vertex **0, 0, 2.3** where 2.3 came after adding vertex and height of **body.trc** such that the neck is placed right on top of the body. The height vector of the cylinder should be **0 0 0.5** and the radius should be **1.4**. Therefore, type:

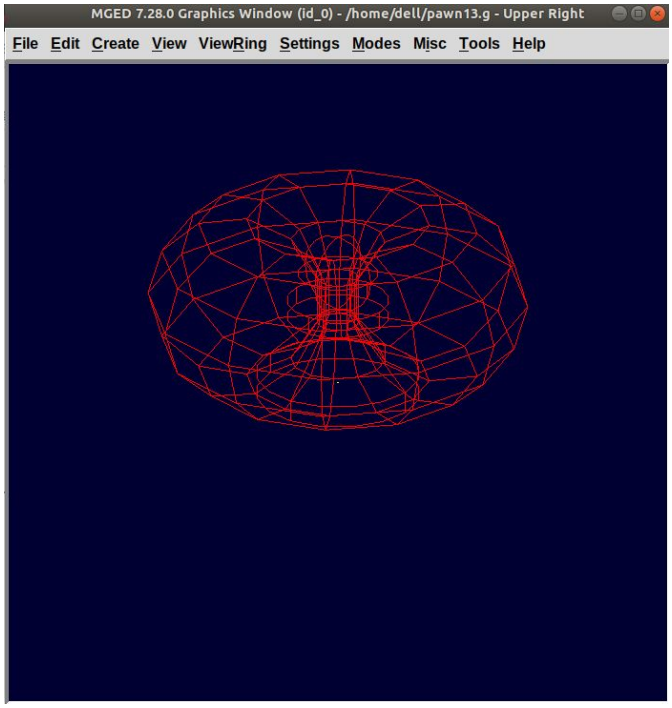
in neck.rcc rcc 0 0 2.3 0 0 0.5 1.4<ENTER>

Making a sphere for the head

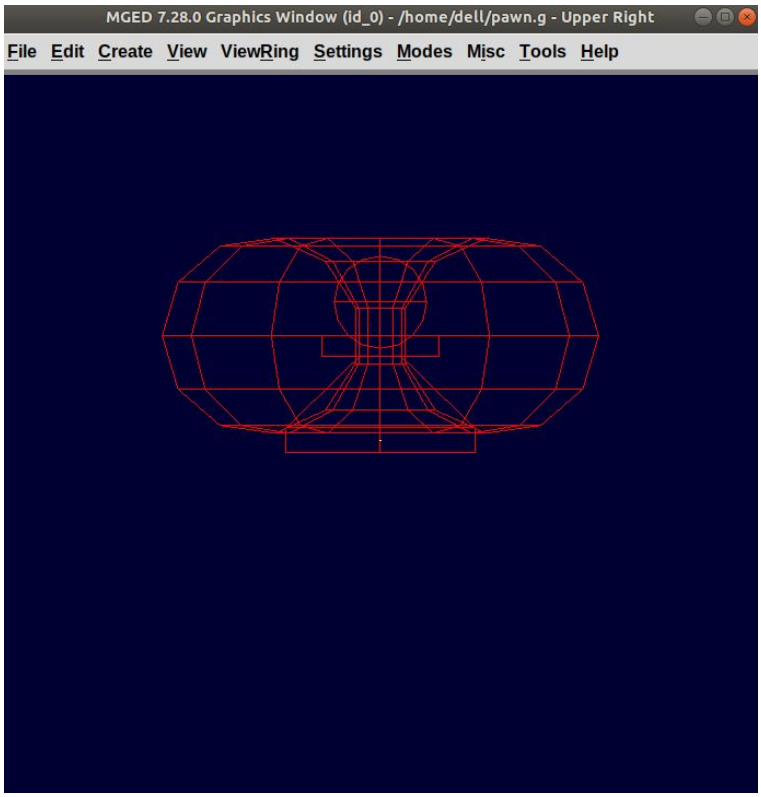
Make a sphere with vertex **0 0 3.6** and radius **1.1**. Technically the vertex of the sphere should be 3.6 i.e., the vertex of neck.rcc (2.3) + half of the height of neck.rcc (0.25) + radius of this sphere (1.1). But we want to cut some portion of the head from below. Type:

in head.sph sph 0 0 3.6 1.1<ENTER>

To zoom out of the view click the left mouse button and to zoom back in click the right mouse button. This is what your pawn looks like till now:



Go to **`View`** from Menu bar and click on **`Front`**. This is what your pawn looks in the front view:



Making a region

Before you can raytrace your design, you have to make of region of all the shapes. Making a region basically means that the shape has uniform material properties i.e., it has mass and occupies space. Constructing a region involves using Boolean operations of union, subtraction, and intersection. To make the region, type:

`r pawn.r u base.rcc u body.trc - curve.tor u neck.rcc u head.sph`<ENTER>

This command tells MGED that
r: Make a region
`pawn.r`: Name it pawn.r
u: Add the volume of the shape
-: Subtract the volume of the shape

Here, we are adding the volume of all the shapes except **`curve.tor`**, which we are subtracting from **`body.trc`** to achieve the required look.

Assigning Material Properties to the Region:

Now type the following in the MGED command window:

`mater pawn.r`

MGED will respond with:

```
Current shader string =
Specify shader. Enclose spaces within quotes.
Shader?
```

MGED asks us to enter the type of material we want our region to be made of. To make the region of plastic. Type in:

`plastic`<ENTER>

Next, MGED will ask for the color. To make our pawn black in color, type:

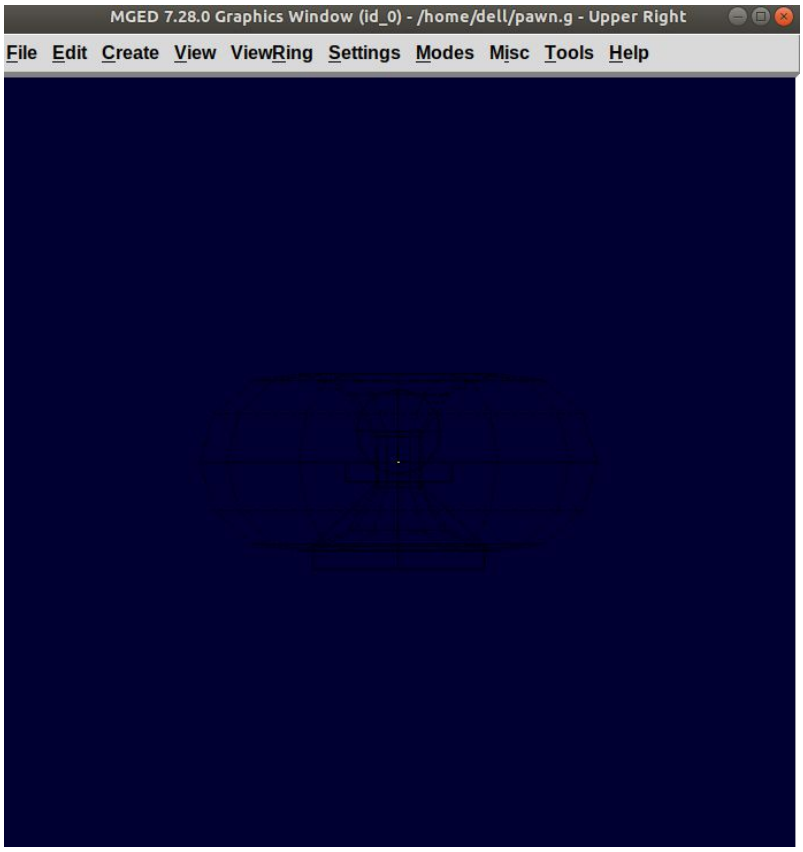
`0 0 0`<ENTER>

At last MGED will ask us if we want to inherit the material properties. To answer with NO, type:

`0`<ENTER>

Clearing the Graphic Window and drawing the new region:
We have shapes visible on our graphics window but it is not our region. To clear the graphics Window of the old design and draw the new region, type:

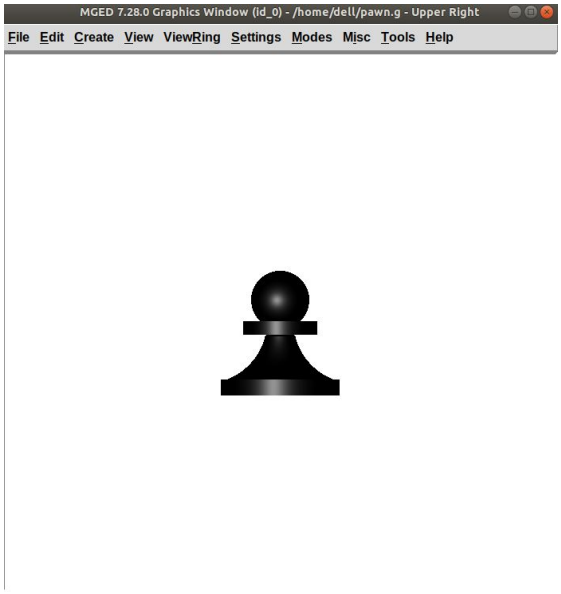
`B pawn.r<ENTER>`



You will see your pawn and the curve.tor is dotted which indicates that it subtracted from the region. This command tells MGED to Blast i.e., clear the graphics window and draw the specified region which in our case is **`pawn.r`**. The Blast command is a combination of **`Z`** and **`draw`** commands. On a side note, draw command is used to draw and existing shape. For example, to draw the sphere you made for the head, type : **`draw head.sph`** which tells MGED to draw head.sph. If the specified shape does not exist, MGED will give an error.

Raytracing your model
Go to the **`File`** menu and select **`Raytrace`**. A dialog box called the **`Raytrace Control Panel`** appears. Next, change the background

color by the raytraced by selecting **`Background Color`**. A dropdown will appear with some predefined color choices and a color tool. Select the white option. To eliminate the wireframing i.e., the outlines of the shapes, go to **`Framebuffer`** (in the Raytrace Panel) and select **`Overlay`**. The display should appear similar to the following illustration:



Your pawn is ready to serve the King. Now it's time to model the rest of the pieces.

8. Modeling Rook

Before you start modeling this piece, create a new database named **`rook.g`**. Create this new database as we did in the previous case.

Type in the Command Window:

opendb rook.g<ENTER>

If you didn't open the MGED Command Window again and used the above command in the already opened window, you will see that the raytraced image didn't disappear. So, in order to get the blue screen back for making other shapes, go to **Modes** from the menu bar and uncheck the **Framebuffer Active** option by clicking on it.

Now, you are ready to model the rook. Since you are already familiar with the **in** command, therefore you will be using the shorthand method of this command for making shapes.

Making the base and body

Making the base is the same as we did in pawn. Type the following in the MGED command window:

in base.rcc rcc 0 0 0 0 0 0.8 2.25<ENTER>

This command will make a cylinder at vertex **0 0 0** with height **0 0 0.8** and radius 2**.25**.

As we did in pawn, we will create the body using two shapes: **rcc** and **trc**. To create the body, type:

in body.trc trc 0 0 0.8 0 0 3 2.25 1.1<ENTER>

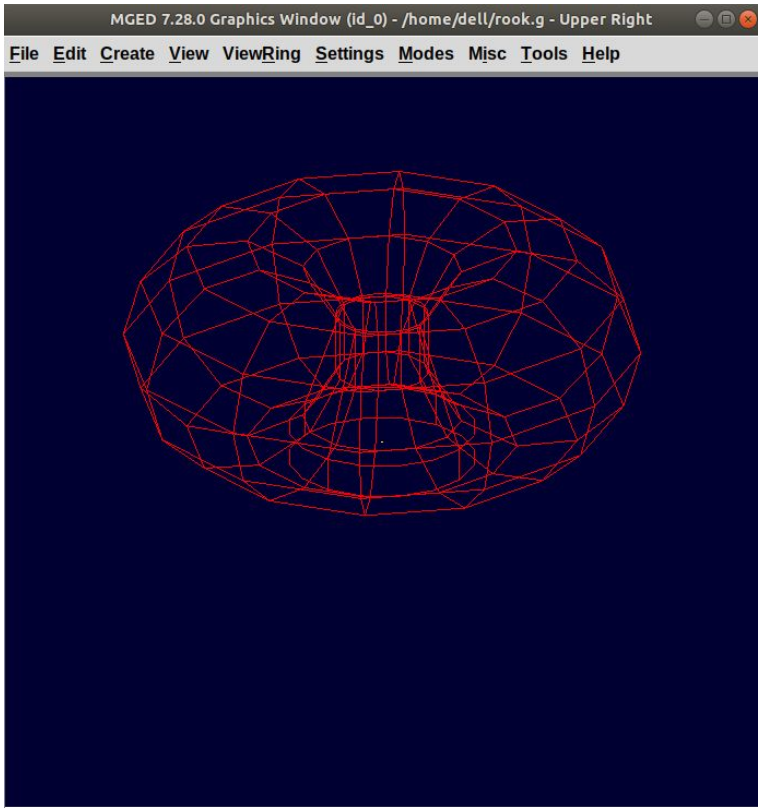
This command creates a trc at vertex **0 0 0.8** with height **0 0 1.5**, radius of the base **2.25** and radius of top **1.1**. Now, to create the curve, type:

in curve.tor tor 0 0 3 0 0 1 3.6 2.6<ENTER>

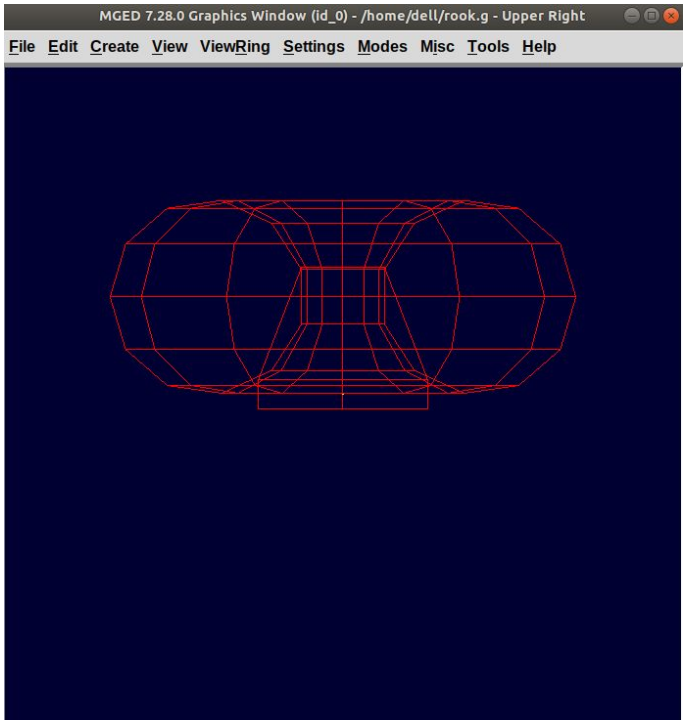
In pawn, we had the vertex at a distance greater than the height of **trc** because we wanted the curve to start right when the body starts but in this case we want to have a straight portion before the curve part. So, we have the vertex at **0 0 3**. The normal vector is **0 0 1** to make our shape perpendicular to z-axis. Radius 1 is **3.6** and Radius 2 is **2.6**.

You will get something like this (after zooming out by clicking the left

mouse button, to zoom in click the right mouse button):



In **Front** view:



Constructing the hollow cylinder for the head

Now comes the tricky part; we need to model the head. To understand it completely, type **Z** to clear the Graphic Window temporarily.

Make sure your Command Window is active while you do so. One of the common mistakes we make as a beginner is that we forget to make the Command Window active and end up typing on the Graphics Window. For those who have typed **Z** but the design started rotating, you need not worry. Go to the **View** option on the Menu bar and click on the last option **Zero**. Now to get back your design in the original orientation, go to **View** option once again and click on the view you were previously in. By default, the view is **az35,el25**. Click on this option and you are ready to move further.

To make a cylinder for neck:

in neck.rcc rcc 0 0 3.8 0 0 1 1.75<ENTER>

The value of vertex **0 0 3.8** came after adding the height of the base and the body. I hope you are familiar with how we use the value of the vertex.

For the head, we have to make a hollow cylinder first, which comes after subtracting a cylinder from another cylinder with a comparatively larger radius. Therefore, the vertex and height of both the inner and outer cylinders should be the same. The radius of the inner cylinder depends on the thickness of the required hollow cylinder.

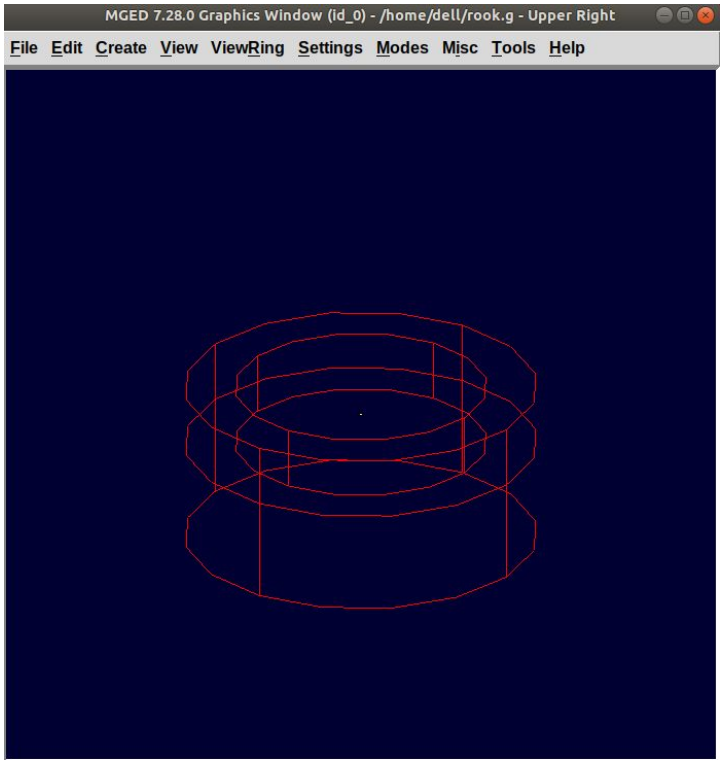
To construct the outer cylinder type:

in outer.rcc rcc 0 0 4.8 0 0 0.6 1.75<ENTER>

To make the inner cylinder with the same vertex and height, type:

in inner.rcc rcc 0 0 4.8 0 0 0.6 1.25<ENTER>

Your graphics window will look like:



The first cuboid for the rook head

Generally, when you see a rook piece its head seems as in a hollow cylinder is cut in pieces. To replicate that, we will make two cuboids with length equal to or greater than the radius of the outer cylinder, and height equal to the height of either one of the cylinders (both inner and outer cylinders have the same height). Then you will subtract these cuboids from the hollow cylinder. Now you will make two cuboids that can be placed perpendicular to each other like an X mark (a cross). For that, we will make **rpp** (Rectangular Parallelopiped).

To make the first one, type:

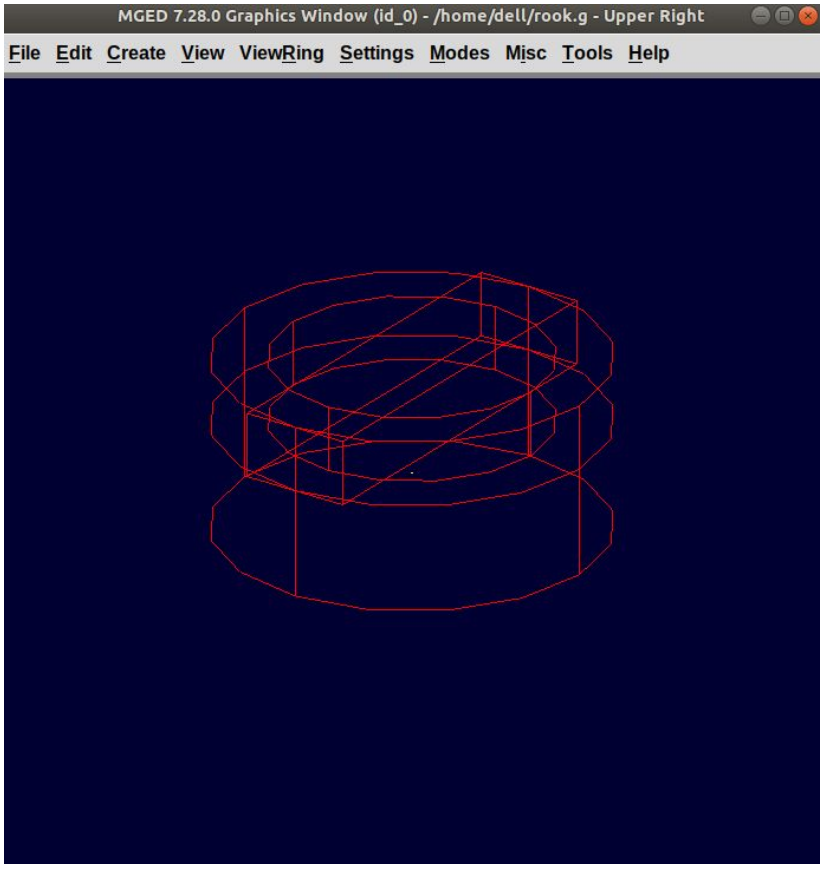
in cross1.rpp rpp<ENTER>

Then MGED will ask for **XMIN**, **XMAX**, **YMIN**, **YMAX**, **ZMIN**, **ZMAX** values. Type:

-1.75 1.75 -0.5 0.5 4.8 5.4<ENTER>

To check the coordinate system, press **m** making sure the Graphics window is active. You won't see the coordinate lines because you are a little above the origin. So, left-click on the graphics window to

zoom out. You will see that the z-axis is along the diameter. Therefore the **XMIN** should be **-1.75** (radius of the outer cylinder) and **XMAX** should be **1.75**. The breadth is along the Y-axis. Therefore, **-0.5** for YMIN and **0.5** for **YMAX**. The height is along the Z-axis. Since the cuboid must start from the base of the outer cylinder, therefore ZMIN is **4.8** and **ZMAX** is **5.4** i.e., **ZMIN** plus height of outer cylinder (**0.6**).



Constructing a cuboid perpendicular to the first

Since you need another cuboid perpendicular to the first one, we use the clone command as follows:

clone -r 0 0 90 cross1.rpp<ENTER>

You are not yet familiar with the clone command which will be explained in detail in the **Modeling Chessboard** section.

Now, MGED will respond with

cross101.rpp {cross101.rpp)

This means we have both shapes for the cross. To view the other shape, type:

draw cross101.rpp<ENTER>

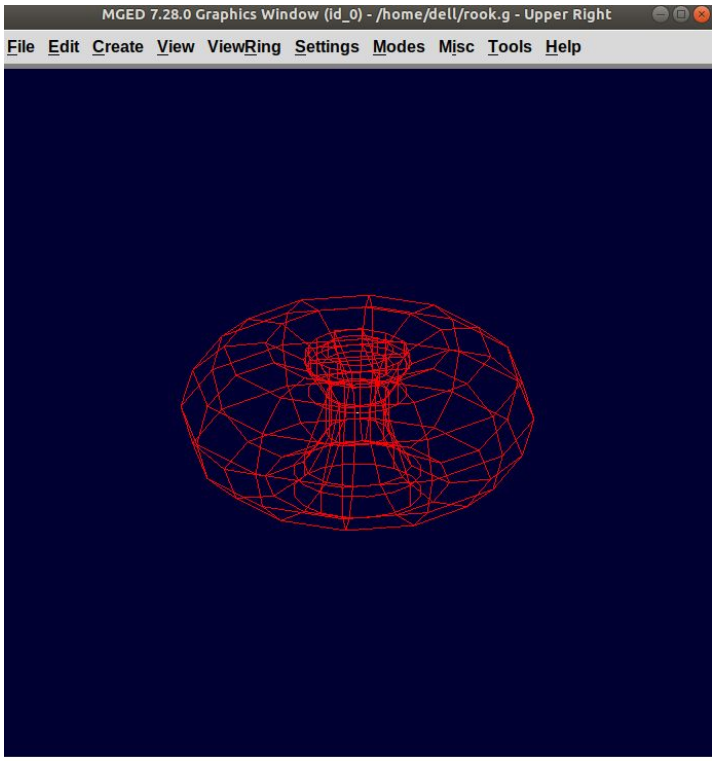
You can look at the head from different views by changing it from the **View** Menu. Don't get discouraged if you only see the head, the other shapes are still right there but since we cleared the Graphics Window using **Z** they are not visible. To get the list of all the shapes in your database, type in the command window:

ls<ENTER>

You will get a list of all your shapes. To view all your shapes on the Graphics Window, use the draw command. Draw all the remaining shapes as follows:

draw base.rcc body.trc curve.tor<ENTER>

Make sure to add spaces between the names. This command tells MGED to draw the three specified shapes. In the az35, el25 view, your design will look like:



Before you raytrace, make the region of the rook:

r rook.r u base.rcc u body.trc - curve.tor u neck.rcc u outer.rcc - inner.rcc - cross1.rpp - cross101.rpp<ENTER>

Here we have subtracted **curve.tor** from **body.trc** to make the curve. Subtracted **inner.rcc** from **outer.rcc** to make a hollow cylinder and subtracted both cuboids **cross1.rpp** and **cross101.rpp** from the outer hollow cylinder to give the finishing look. This command makes a region named **rook.r**.

Assigning material properties and raytracing

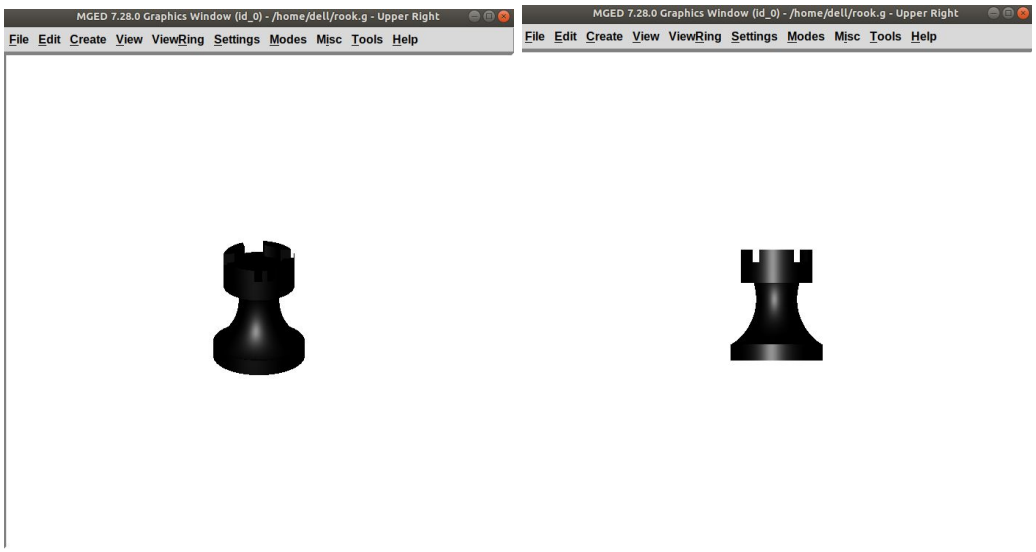
We will assign material properties as we did in the case of pawn. We will use the shorthand method of the mater command. Type:

mater rook.r plastic 0 0 0 0<ENTER>

Don't forget to clear the graphics window and redraw the design using Blast command as follows:

B rook.r<ENTER>

Now, raytrace your design from the **File** menu. Change the background color to white and select the **Overlay** option from **Framebuffer** option in the Raytrace Menu Bar. For details check the instructions in the previous model of the pawn. This is what we get after raytracing:



az35, el25 view **Left view**

9. King:

Now it's time to model the king. It is comparatively easier than the above pieces. To begin modeling, create a new database, type in the command prompt:

opendb king.r<ENTER>

You have your Command and Graphics ready after confirming in the dialog box.

If you look at all the chess pieces, you see that the base and body of almost all the pieces are the same and they only differ in the head area.
To make the base and the body, type in:

in base.rcc rcc 0 0 0 0 0 0.7 2.25<ENTER>

in body.trc trc 0 0 0.7 0 0 2.2 2.25 0.85<ENTER>

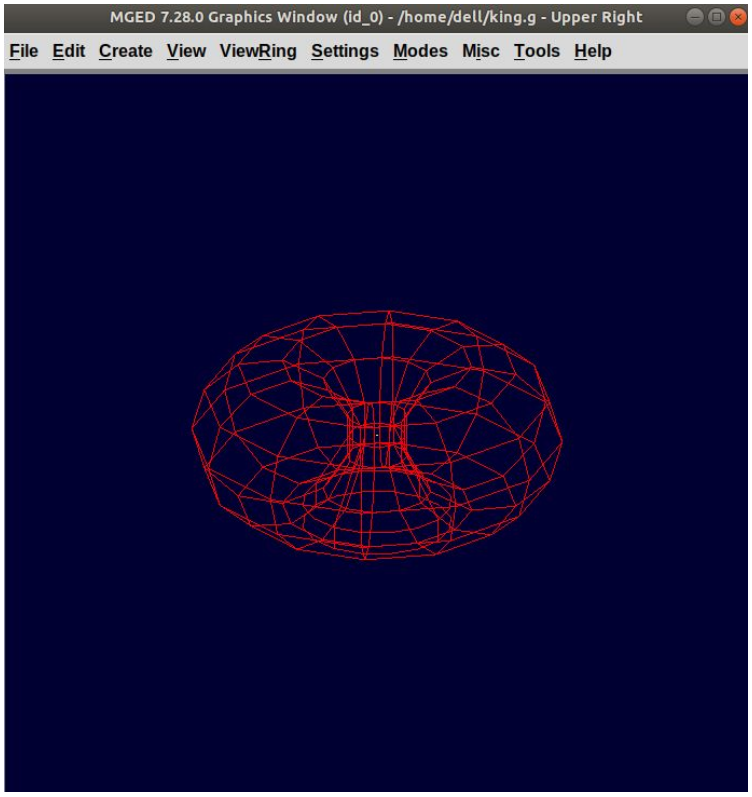
in curve.tor tor 0 0 2.9 0 0 1 3.2 2.4<ENTER>

To make a cylinder for the neck, type:

in neck.rcc rcc 0 0 2.9 0 0 0.5 1.4<ENTER>

As described in the above pieces, the shape neck.rcc must be placed

at the top of body.trc. Once we have made the base, body, and curve we get something like this:



When we look closely at the end product, the head can be divided into three parts, the head bottom, the middle section, and the tiny top section. All of these are trc. To make the bottom part of the head, type:

in headbottom.trc trc 0 0 3.4 0 0 1.5 0.8 1.4<ENTER>

Here the base radius of the **headbottom.trc** is equal to the top radius of body.trc.

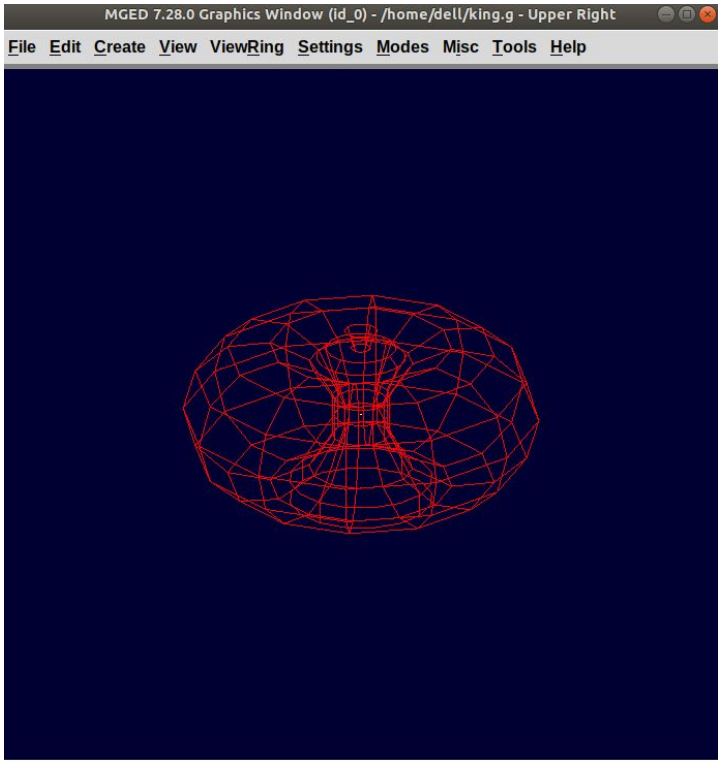
Since the top radius of **headbottom.trc** should be equal to the base radius of the headmid.trc. Therefore, to make the mid part, type:

in headmid.trc trc 0 0 4.9 0 0 0.3 1.4 1.1<ENTER>

Now to make the top part this head, type:

in headtop.trc trc 0 0 5.2 0 0 0.6 0.3 0.5<ENTER>

After this, your Graphics Window looks like:



You have all your shapes now. It is time to make a region of it:
r king.r u base.rcc u body.trc - curve.tor u neck.rcc u headbottom.trc u headmid.trc u headtop.trc<ENTER>

Now, assign material properties using the following command:
Type:
mater king.r plastic 0 0 0 0<ENTER>

Before raytracing, use the blast command as follow:
B king.r<ENTER>

To achieve the target design, change the view to **Front** from the **View** menu. Now raytrace your design from the **File** menu.

10. Queen:

As always, begin by creating a new database using the following command in the command prompt:

opendb queen.r<ENTER>

The King piece and the queen differ only in the top part. So, we will

reuse the commands we used in the upper section. Type in the Command Window:

in base.rcc rcc 0 0 0 0 0 0.7 2.25<ENTER>

in body.trc trc 0 0 0.7 0 0 2.2 2.25 0.85<ENTER>

in curve.tor tor 0 0 2.9 0 0 1 3.2 2.4<ENTER>

in neck.rcc rcc 0 0 2.9 0 0 0.5 1.4<ENTER>

If you look closely, the only difference is the height of the **headmid.trc** and the top section of the queen is a sphere. So, type:

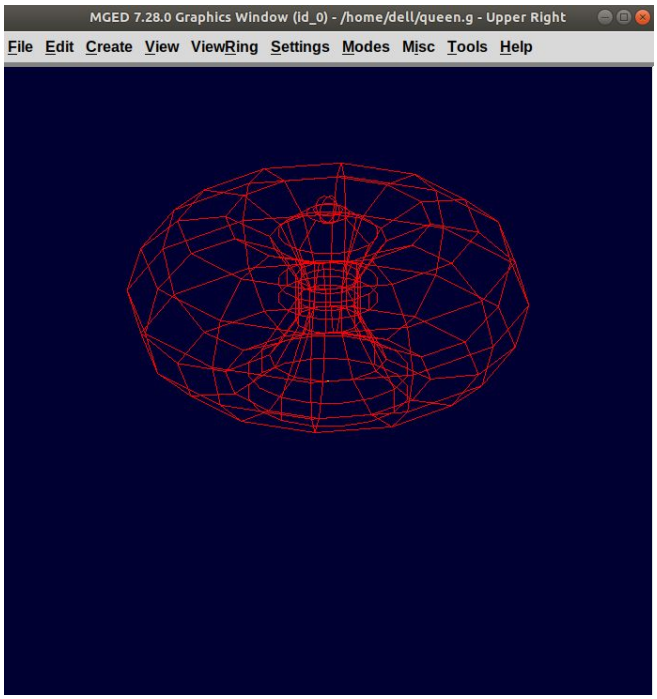
in headbottom.trc trc 0 0 3.4 0 0 1.5 0.8 1.4<ENTER>

in headmid.trc trc 0 0 4.9 0 0 0.6 1.4 0.6<ENTER>

in headtop.sph sph 0 0 5.6 0.4<ENTER>

The portion of the sphere at the top is slightly larger in size than a semi-sphere. So, the vertex of the sphere is **0 0 0.5** i.e., vertex of **headmid.trc** + height of **headmid.trc** + **0.1**. The value 0.1 is added to make it slightly larger than a semi-sphere.

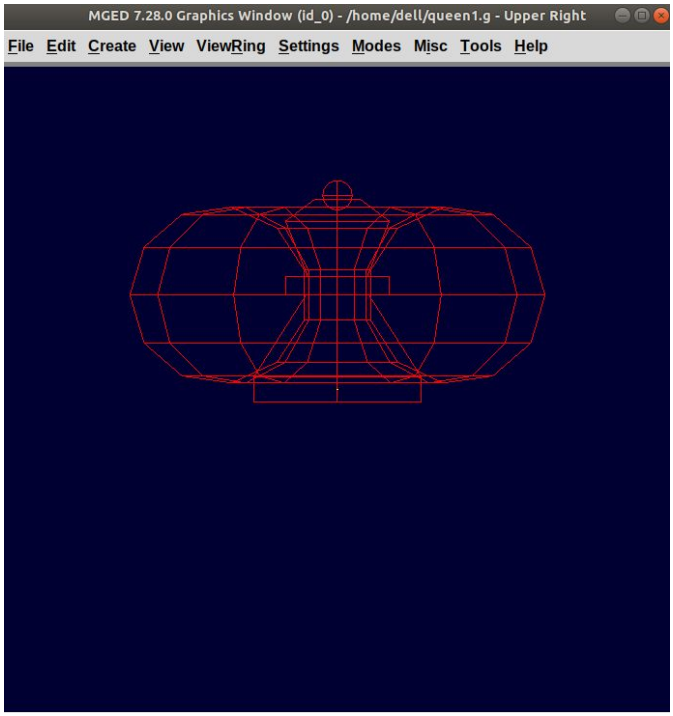
The output is:



Make a region using the following command:

r queen.r u base.rcc u body.trc - curve.tor u neck.rcc u headbottom.trc u headmid.trc u headtop.sph<ENTER>

The front view looks like:



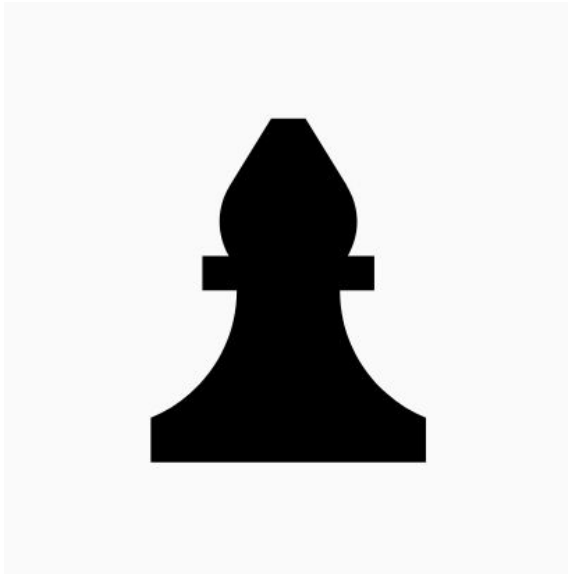
Now, comes the part of assigning the material properties and finally Raytracing the design. Type:

mater queen.r plastic 0 0 0 0<ENTER>

B queen.r<ENTER>

After Raytracing, the queen in **`Front`** view looks like:

11. Bishop:



Begin by creating a new database, name it **`rook.g`**.

Using the same commands for the base:

in base.rcc rcc 0 0 0 0 0 0.7 2.25<ENTER>

in body.trc trc 0 0 0.7 0 0 2.2 2.25 0.85<ENTER>

in curve.tor tor 0 0 2.9 0 0 1 3.2 2.4<ENTER>

in neck.rcc rcc 0 0 2.9 0 0 0.5 1.4<ENTER>

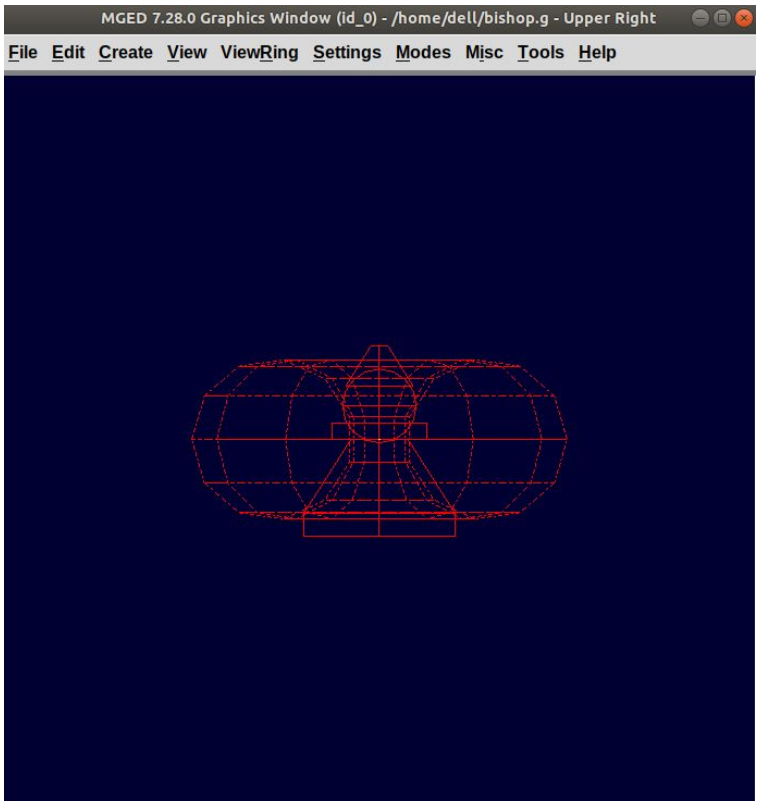
Now, coming to the head of the bishop, you will use two shapes for it, **sph** and **trc**.
The sphere starts from the base of **neck.rcc**, therefore the vertex of this sphere equals to vertex of **neck.rcc** + radius of the sphere i.e., **2.9** + **1.1** = **4**. Type:

in head.sph sph 0 0 4.0 1.1<ENTER>

The top has vertex **0 0 4.4**, where **4.4** = z value vertex of **head.sph** (**4.0**) + (**0.4**)

in headtop.trc trc 0 0 4.4 0 0 1.2 1 0.25<ENTER>

Since you have got all the shapes, you have an output like this in the **Front** view:



Make the region:

r bishop.r u base.rcc u body.trc - curve.tor u neck.rcc u head.sph u headtop.trc<ENTER>

Assign material properties:

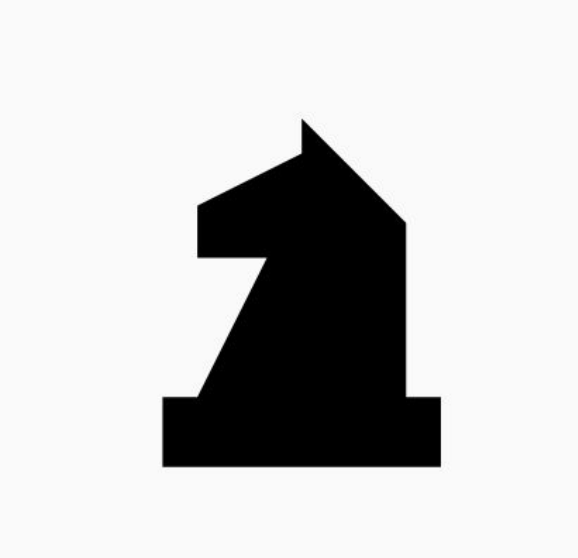
mater bishop.r plastic 0 0 0 0<ENTER>

Before you Raytrace, don't forget to blast your region using the command

B bishop.r<ENTER>

Now Raytrace it with a white background. The front view after we raytracing looks like this:

12. Knight



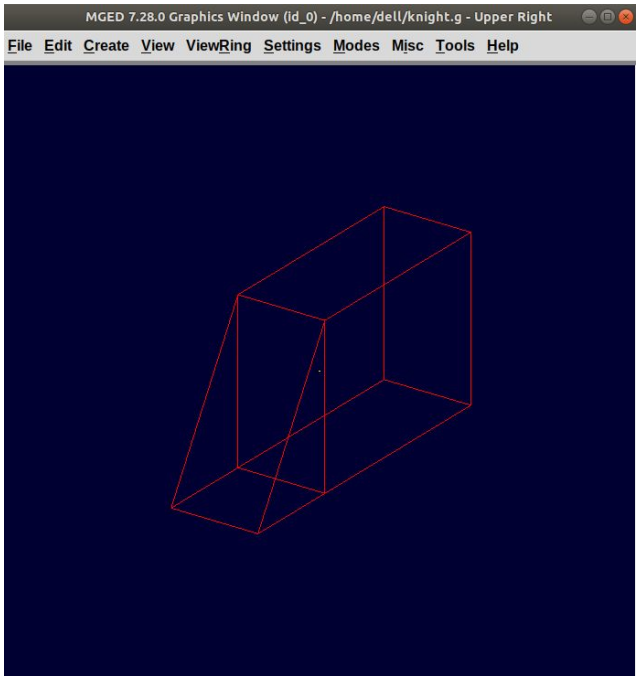
Last but not least, it is time to model the knight. I hope till now you are a little confident while working with dimensions because this section is going to have plenty of measurements.

Begin by creating a new database named **rook.g**.

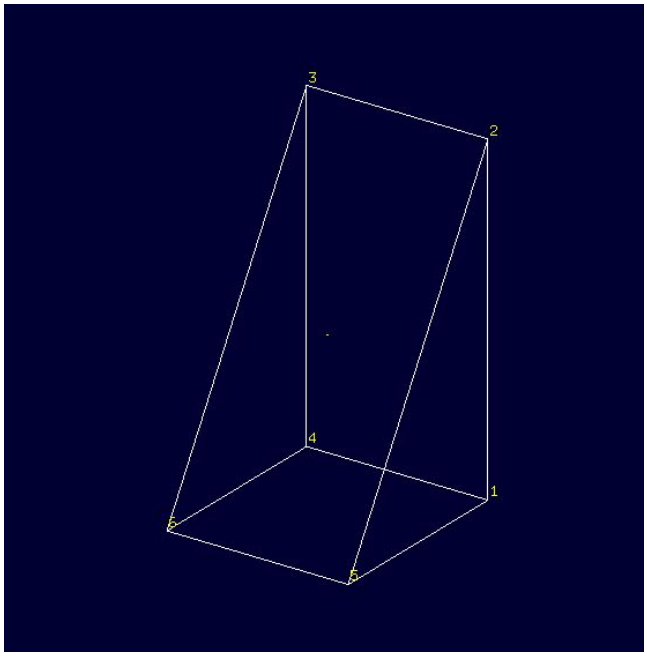
The Knight piece can be broken down into four sections: base, body, neck and the top.
Starting with the base which is same as the other pieces, type in the MGED command window:

in base.rcc rcc 0 0 0 0 0 1.1 2.25<ENTER>

Now, coming to the body. The body section is made up of two shapes; **arb6** and **rpp.** You are already familiar with **rpp(Rectangular Parallelepiped)**, so let's get you introduced with **arb6 (Arbitrary Convex Polyhedron, 6pts)** You will use a shape like the one given below:



While making this shape using the **in** command, MGED will ask you to enter the values of all six points. The following image gives an idea of the points:



You will use this shape to make the left part of the body section. To insert this shape, type:

in body1.arb6 arb6<ENTER>

We will the same as below:
MGED will then ask you to enter x, y, z values of all six points, one by one. Let's understand each point and its value.
For **point 1**, type
0.65 0.5 1.1<ENTER>

For **point 2**, type
0.65 0.5 2.9<ENTER>

For **point 3**, type
0.65 -0.5 2.9<ENTER>

For **point 4**, type
0.65 -0.5 1.1<ENTER>

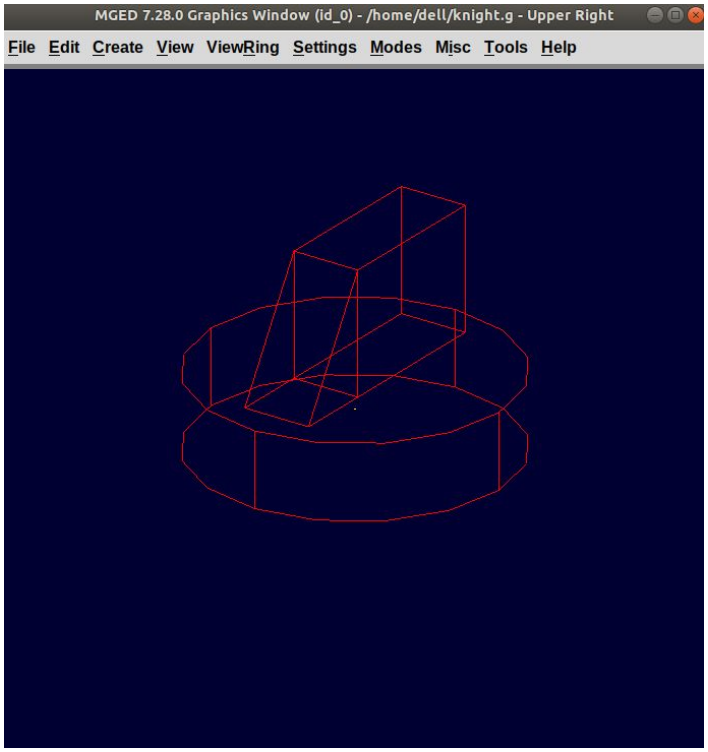
For **point 5**, type
1.75 0.5 1.1<ENTER>

For **point 6**, type
1.75 -0.5 1.1<ENTER>

Here, 1.75 = radius of base.rcc (2.25) - the distance of the body from the edge of base (0.5)
0.5 = half of body's width
1.1 = height of base.rcc
2.9 = height of base.rcc (1.1) + height of body (1.8)

To make the other part of the body, type:
in body2.rpp rpp -1.75 0.65 -0.5 0.5 1.1 2.9<ENTER>

This is what we get as output:



Moving on to the neck, it also consists of two parts. You will make two rpp. As you look at the target design, the left side of the neck has a slightly greater height than the right side. So, to make two rpp of different heights, type:

in neck1.rpp rpp 0 1.75 -0.5 0.5 2.9 3.7<ENTER>

in neck2.rpp rpp -1.75 0 -0.5 0.5 2.9 3.4<ENTER>

The top also has two parts, left and right arb6. The left one starts from the top of neck1.rpp and the right one starts at the top of neck2.rpp. Also, the right arb6 has a height slightly greater than the left one. To get the shapes, type:

in top1.arb6 arb6<ENTER>

0 0.5 3.7<ENTER>

0 0.5 5.2<ENTER>

0 -0.5 5.2<ENTER>

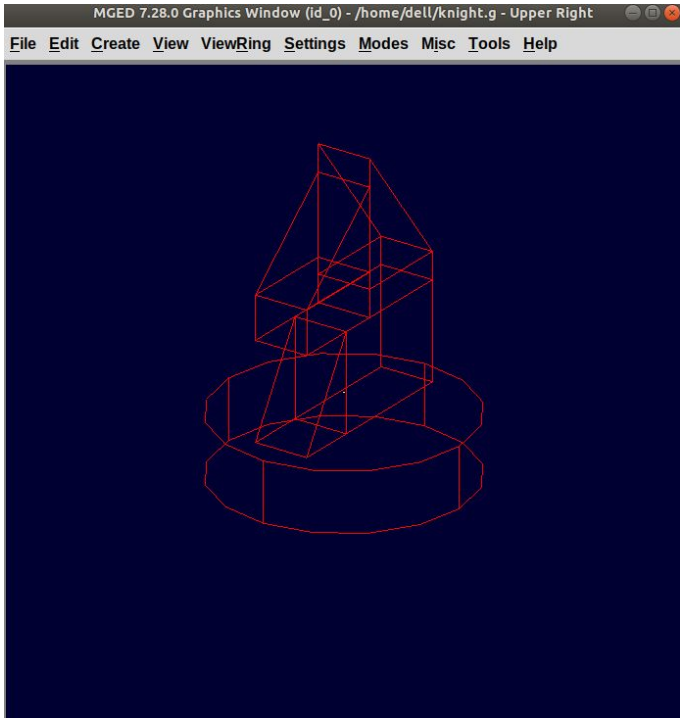
0 -0.5 3.7<ENTER>

1.75 0.5 3.7<ENTER>

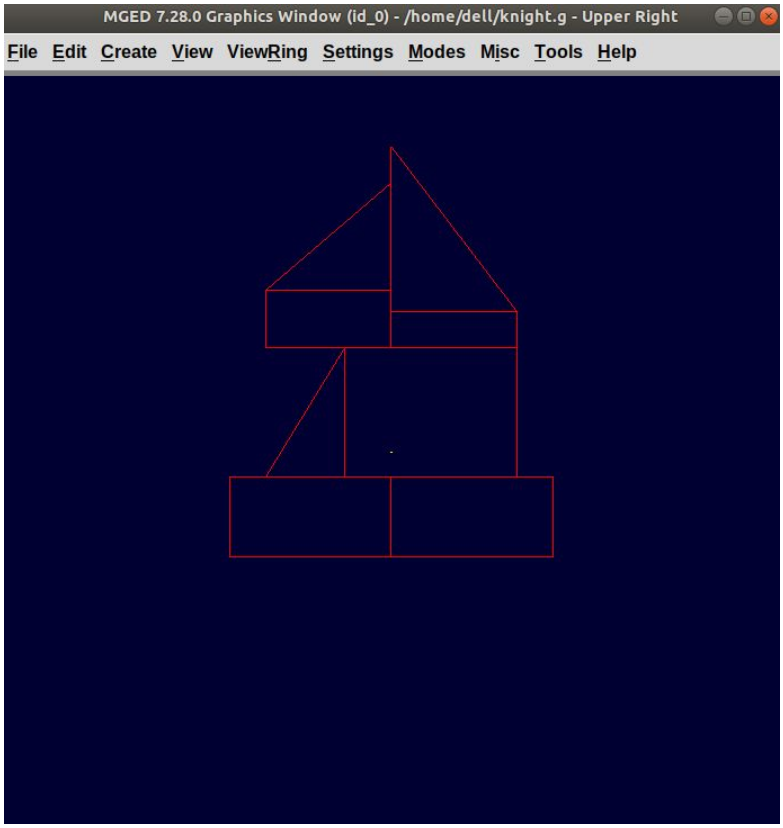
1.75 -0.5 3.7<ENTER>

in top2.arb6 arb6<ENTER>
-1.75 0.5 3.4<ENTER>
0 0.5 5.4<ENTER>
0 -0.5 5.4<ENTER>
-1.75 -0.5 3.4<ENTER>
0 0.5 3.4<ENTER>
0 -0.5 3.4<ENTER>

On a side note, these commands can be written in the shorthand method with all the values in a single row separated by spaces. Our Graphics Window looks like this:



And in Left view:



Type the following command to make the region:

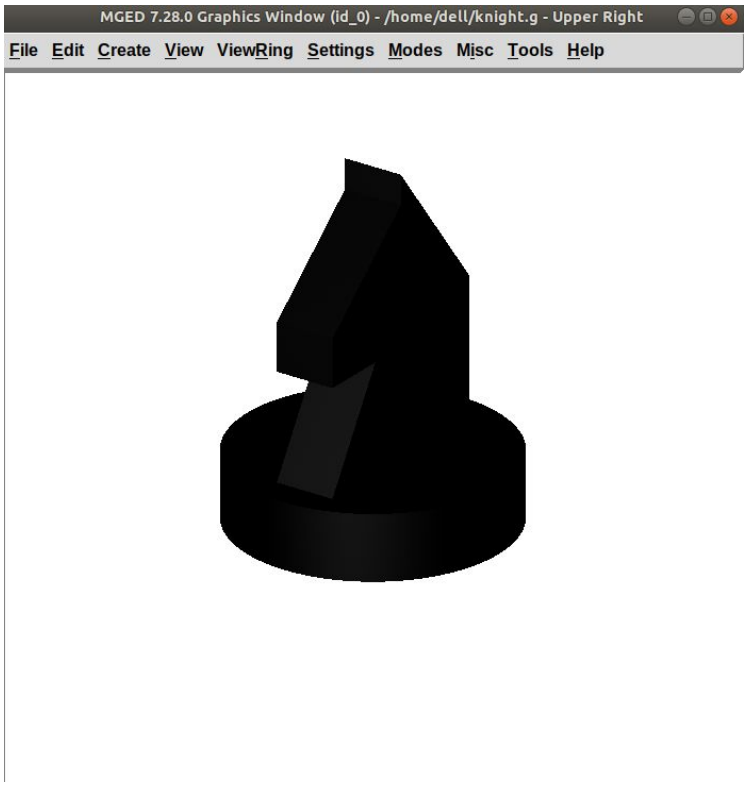
r knight.r u base.rcc u body1.arb6 u body2.rpp u neck1.rpp u neck2.rpp u top1.arb6 u top2.arb6<ENTER>

Now, assign the material properties to this knight and redraw your design. Type:

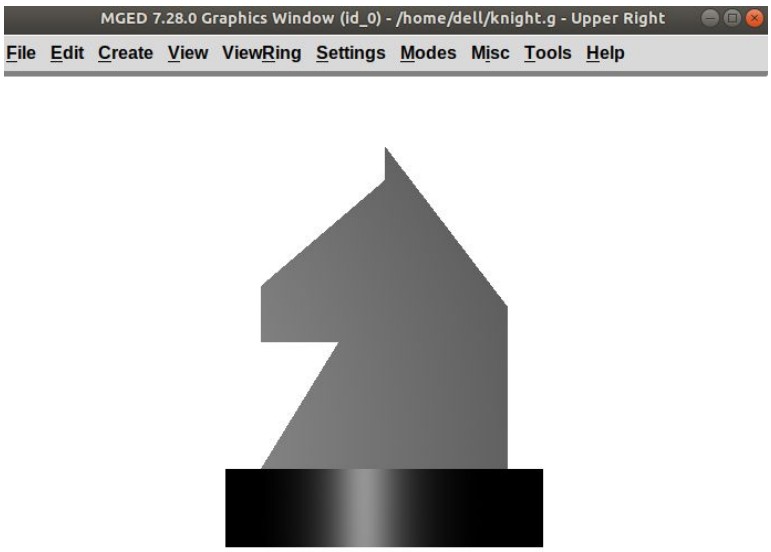
mater knight.r plastic 0 0 0 0<ENTER>

B knight.r<ENTER>

After raytracing your design looks like:


MGED 7.28.0 Graphics Window (id_0) - /home/dell/knight.g - Upper Right
File Edit Create View ViewRing Settings Modes Misc Tools Help

In Left view:

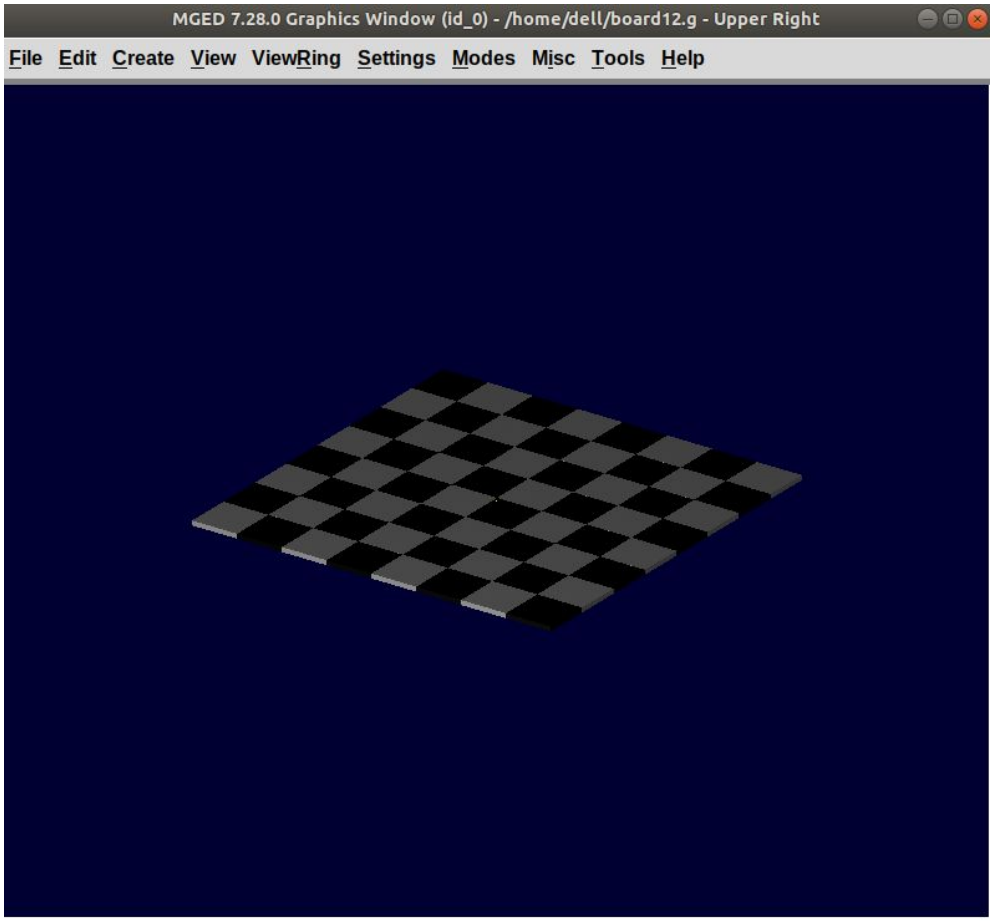

MGED 7.28.0 Graphics Window (id_0) - /home/dell/knight.g - Upper Right
File Edit Create View ViewRing Settings Modes Misc Tools Help

There is a color difference because the part above is plain and the bottom area is round.

13. Modeling the Chess Set

Chessboard:

In this lesson, we are going to model the chessboard. Given below is our target design:



Create a new database name **chess.g**

opendb chess.g<ENTER>

Assigning a title to your database

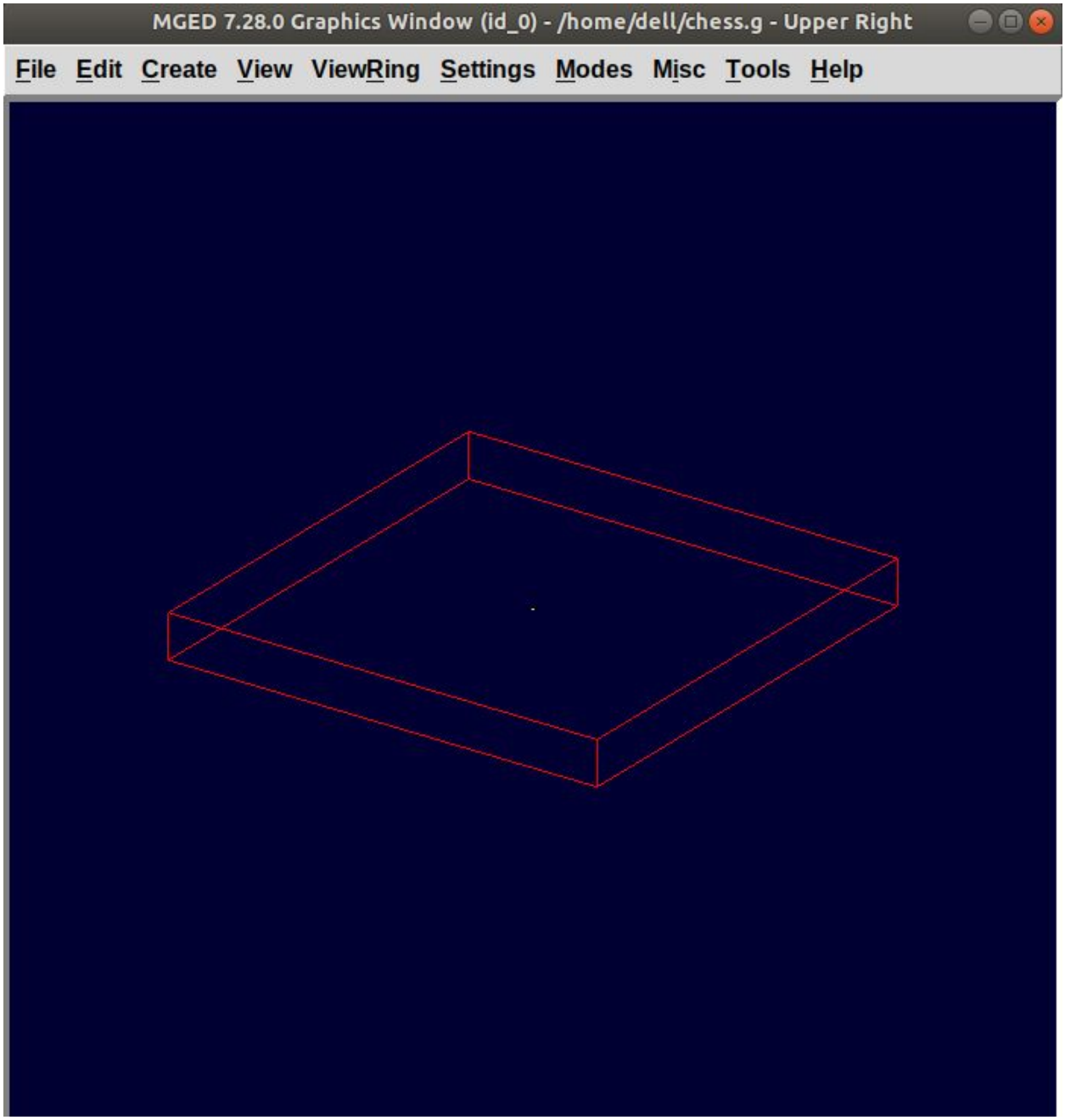
In the MGED command window, type **title** followed by the title of your database justifying what you are making. Press **Enter** at the end.

mged>title Chess<ENTER>

Creating a single tile for the chessboard

Before beginning, make sure that MGED Command Window is active (by clicking anywhere in the window). Then type in the command:

```
in tile.s1 rpp 0 1 0 1 -0.1 0
```



Making a region of the tile

Type in the MGED prompt:

`r tile.r1 u tile.s1`

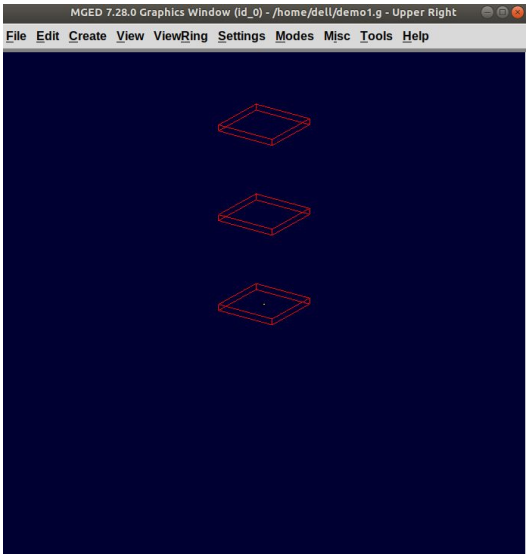
This command makes a region with the name **`tile.r1`**

- **Understanding the clone command**

Let's understand the clone command first:
This command is used to do deep copying in MGED. The syntax for this command is:
clone [-abhimnprtv] <object>
Here each one of the **[-abhimnprtv]** have a specific meaning to it. Let's better understand this command using examples.

Create a demo database by typing **mged demo.g** in the command prompt. Then, create a sphere using the **in** command as follows:
in tile.s1 rpp 0 1 0 1 -0.1 0<ENTER>

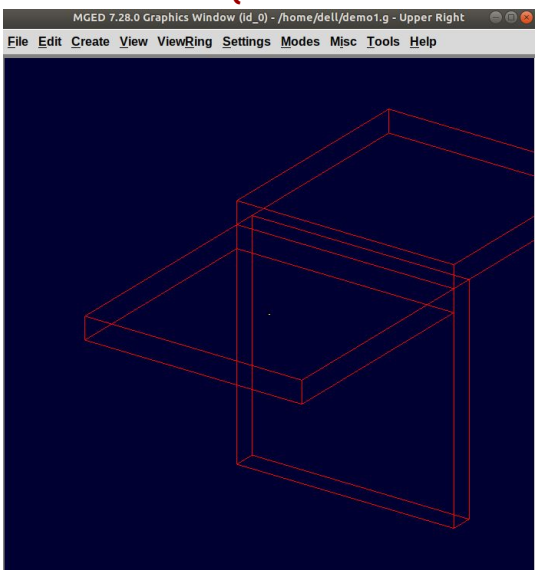
-a <n> <x> <y> <z> Specifies the translation split dimensions between n clones.
Type,
clone -a 2 0 0 3 tile.s1<ENTER>
MGED will show this:
tile.s101 {tile.s101 tile.s201}
This means, you have got two clones separated by distance 3 units on the z -axis. To visually verify it, type:
draw tile.s101 tile.s201<ENTER>



-b <n> <x> <y> <z> Specifies a rotation around the x, y and z axes split between n copies
Example: Type **Z** on the MGED command prompt to clear the

graphic window.
Then type,

```
mged>draw tile.s1
mged>clone -b 2 0 180 0 tile.s1
tile.s301 {tile.301 tile.401}
```



You see that the cloned tiles are separated from each other by an angle of 180 along the y axis.

-c Increment the second number in object names.
-f Don't draw the new object.
-g Don't resize the view after drawing new objects.
-h Prints the message.
-i <n> Specifies the increment between each copy.
-m <axis> <pos> Specifies the axis and point to mirror the group.
-n <# copies> Specifies the copies you make.
-p <x> <y> <z> Specifies point to rotate around for -r. Default is 0 0 0.
-r <x> <y> <z> Specifies the rotation around x, y and z axes. It works same as **-b** when combined with **-n**.
-t <x> <y> <z> Specifies the translation between each copy. It works same as **-a** when combined with **-n**.
-v Prints version info.

- **Cloning the tile for three other tiles**

Coming back to our chessboard, let's clone the tile we made earlier. Now, type:

```
mged>clone -t 2 0 0 -i 1 -n 3 tile.r1<ENTER>
```

MGED will respond with:

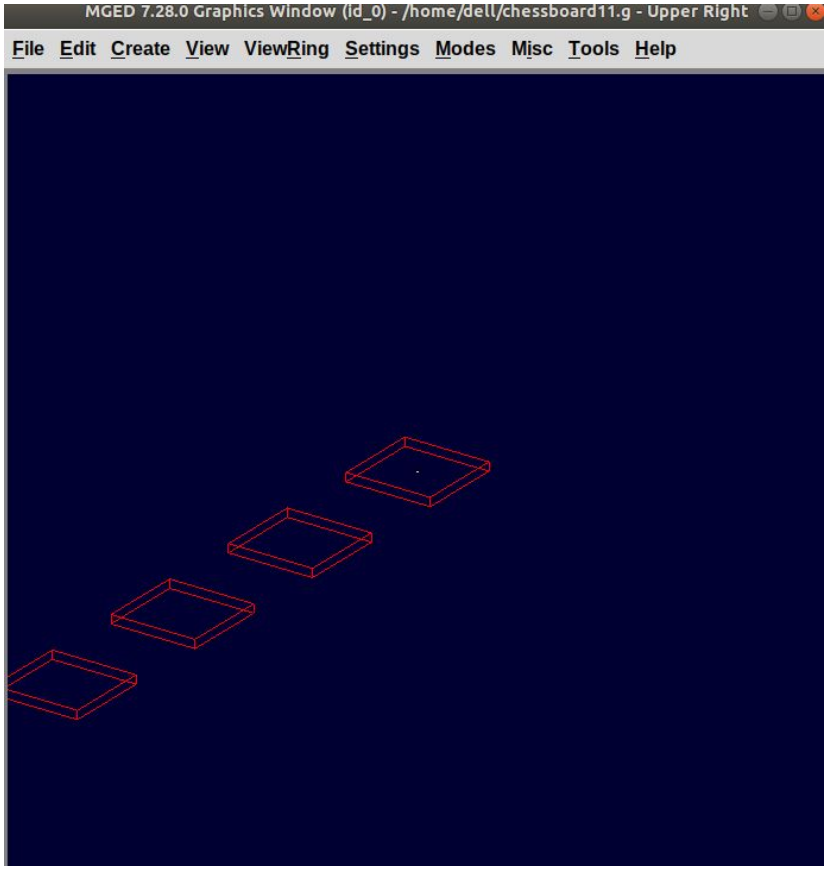
```
tile.r2 {tile.s2 tile.s3 tile.s4 tile.r2 tile.r3 tile.r4}
```

This command translates the first clone tile.r2 at x:2 y:0 z:0 (leaving a gap of 1 unit) and then increments it value by 1 for the next clone. **-n** 3 specifies that it creates three clones (tile.r2 tile.r3 tile.r4)

This command successfully creates single colored tiles of a row. Till now, you can only see one tile in the graphics window, to see all the clone, type:

```
mged>draw tile.r2 tile.r3 tile.r4<ENTER>
```

Left click on the graphic window for all the tiles to fit in the graphics window.



Grouping the tiles in a row

Let's group these tiles in one row:

```
mged>g row.g1 tile.r1 tile.r2 tile.r3 tile.r4<ENTER>
```

Creating an alternate row by cloning

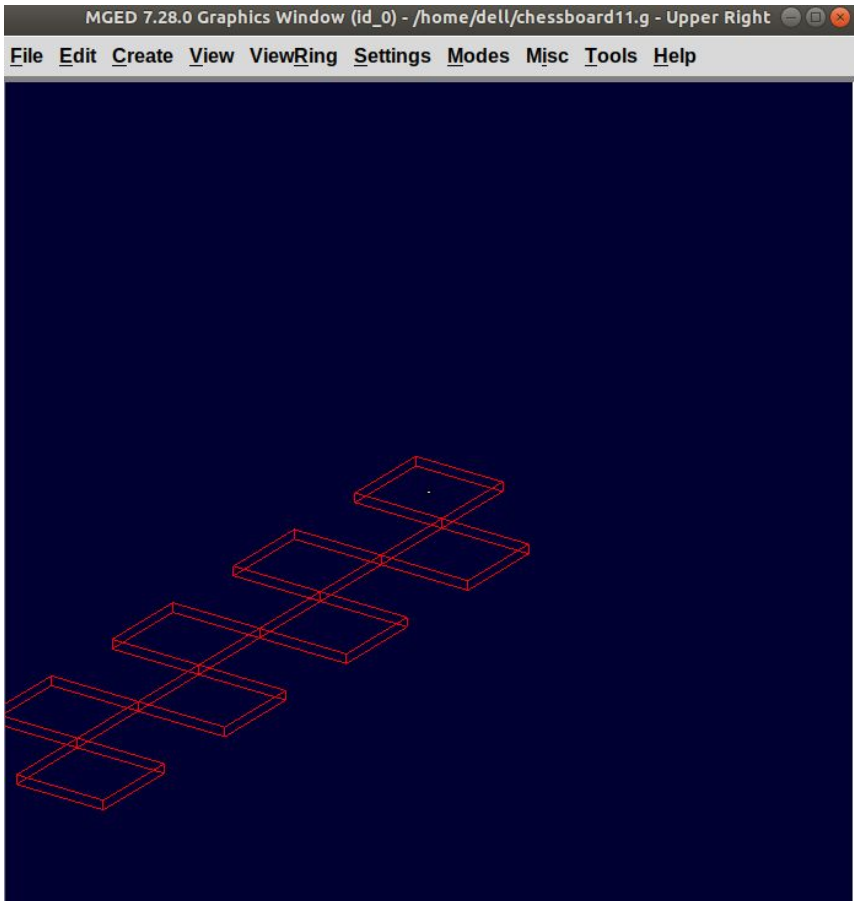
To get the tiles in alternate position as present in the above screenshot, we will clone this row and translate it to x:1 y:1 z:0, as shown below:

```
mged>clone -t 1 1 0 -i 1 row.g1<ENTER>
```

MGED will respond with:

```
row.g2 {tile.s5 tile.r5 tile.s6 tile.r6 tile.s7 tile.r7
tile.s8 tile.r8 row.g2}
```

Type **draw row.g2<ENTER>** in the command prompt to view the row.g2:



There are total 8 rows in a chess board, so we will have 3 clones each of **row.g1** and **row.g2**

Making the remaining rows

Now, we will be translating along the y axis, therefore the command will be:

```
mged>clone -t 0 2 0 -i 1 -n 3 row.g1<ENTER>
```

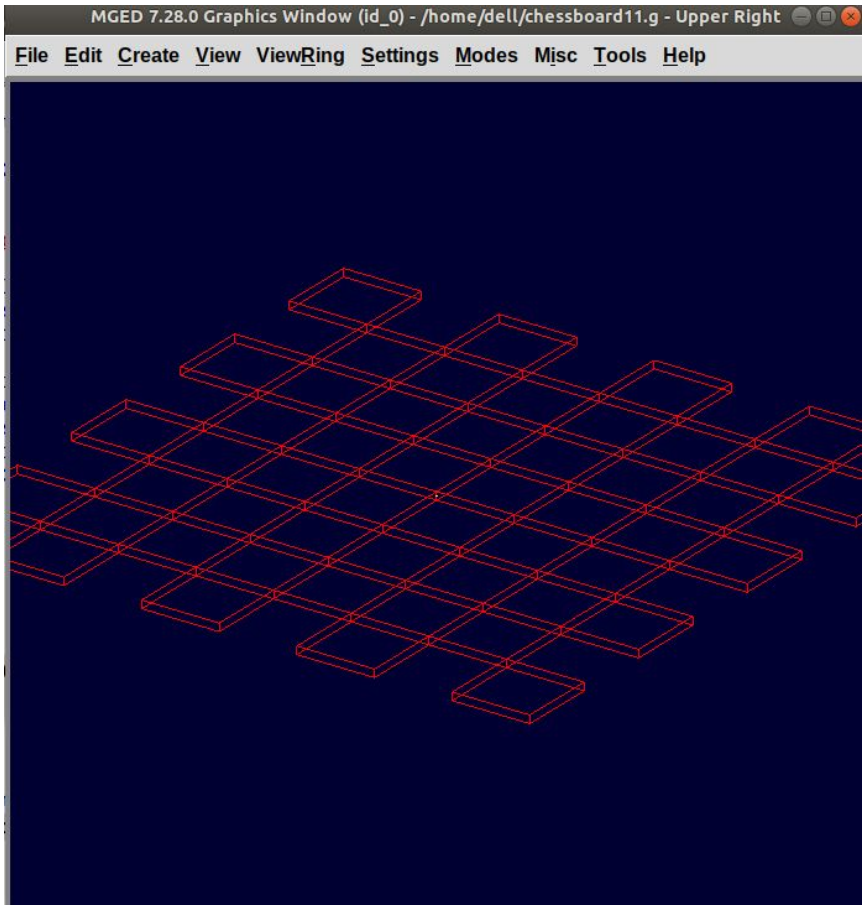
And for row.g2

```
mged>clone -t 0 2 0 -i 1 -n 3 row.g2<ENTER>
```

You will get **row.g3**, **row.g4**, **row.g5** clones of **row.g1** and clones **row.g6**, **row.g7**, **row.g8** of **row.g2**

You can view these rows by using the draw command:

draw row.g3 row.g4 row.g5 row.g6 row.g7 row.g8<ENTER>



You see that here we only have alternate tiles i.e., tiles of one color. Let's group them together under one name **black.g** as shown below:

mged>g black.g row.g1 row.g2 row.g3 row.g4 row.g5 row.g6 row.g7 row.g8<ENTER>

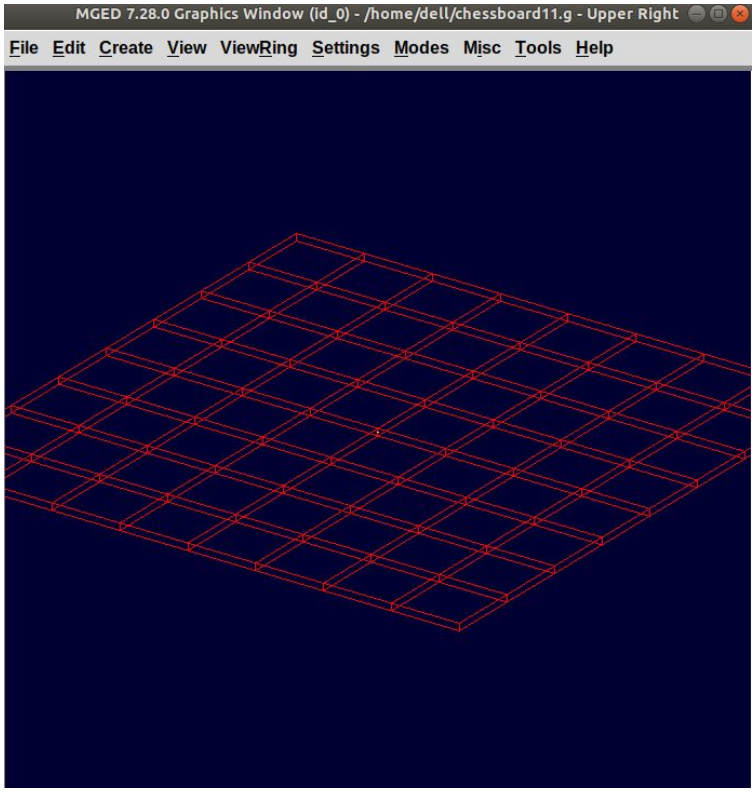
Now, in order to make the white tiles we will clone the group of black tiles **black.g**, as shown below:

mged>clone -r 0 0 90 -p 4 4 0 black.g<ENTER>

We will get a clone with the name black.g2. We can change the name of this group to white.g using the **mv** command.

mged>mv black.g2 white.g<ENTER>

Now, let's have a look at the rest of our slides by typing **draw white.g**



Giving colors to our tiles:
In this section we will color our tiles using the **comb_color** command which means combination color i.e. color of the whole combination/group. The syntax to use this command is:
comb_color combination R G B
Where **combination** is the name of the combination we want to color. **R**, **G** and **B** are the red, green and blue values respectively.

To color the black tiles:

```
mged>comb_color black.g 0 0 0<ENTER>
```

To color the white tiles:

```
mged>comb_color white.g 255 255 255<ENTER>
```

Now combine these black and white tiles to form a board.

```
mged>g board.g black.g white.g<ENTER>
```

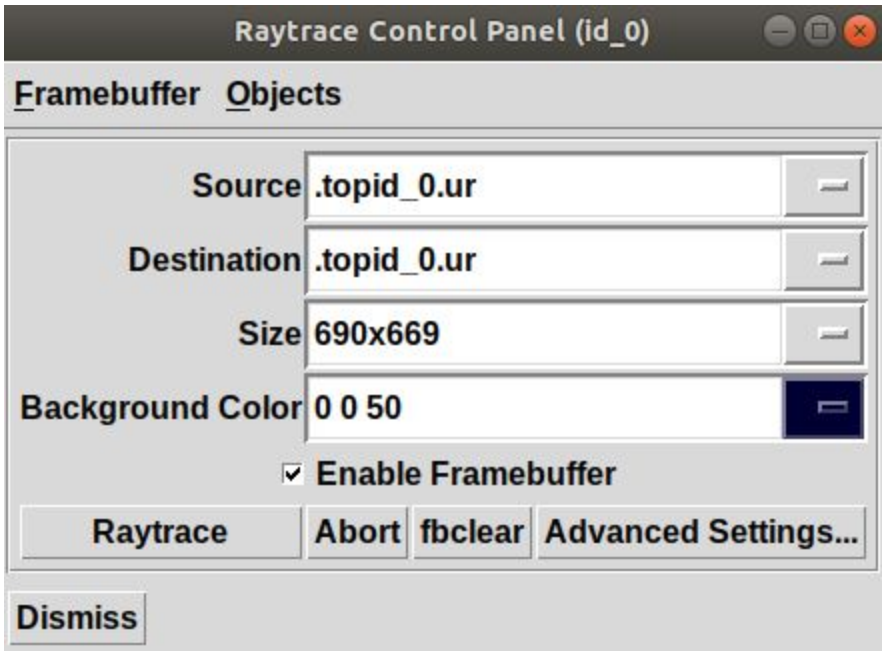
And then comes the final part, which is to raytrace our chessboard. Before we raytrace, move the mouse pointer to the Command Prompt and type at the prompt:

`B board.g<ENTER>`

This command clears the screen and redraws the board with the specified colors.

- Raytracing the Board

Go to the **`File`** menu and select **`Raytrace`**. The Raytrace Control Panel opens. To have a lighter background, click on the dropdown button on the right of **`Background Color.`** Click on **`Raytrace`** to start the raytracing process.



While it is raytracing, click on the Framebuffer options in the **`Raytrace Control Panel Menu Bar`** and click on **`Overlay`**.

After the raytracing process is completed, you get a board, as shown below: