

PROGRAMMING  
HERSHEY 3D  
PROGRAMMING  
HERSHEY 3D  
PROGRAMMING  
HERSHEY 3D  
PROGRAMMING  
HERSHEY 3D  
PROGRAMMING  
HERSHEY 3D

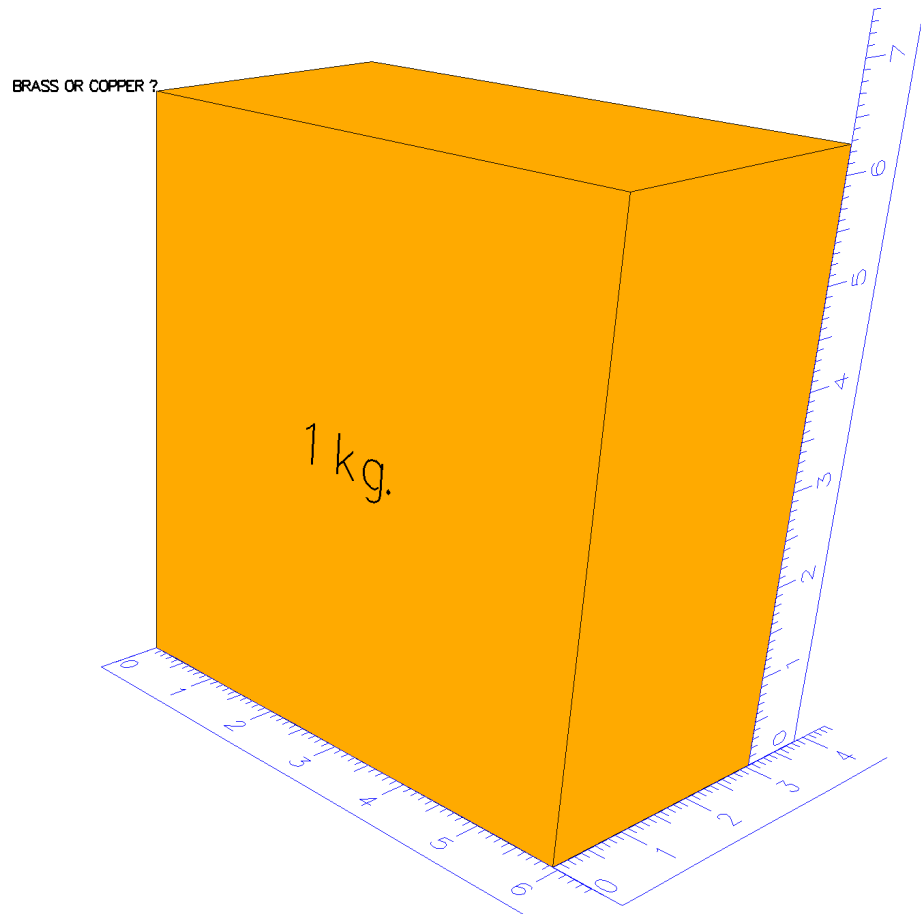
## Sketch3D.pl

This builds upon the previously-discussed Sketch package which builds around the Hershey fonts to create a drawing in the horizontal and vertical dimensions (2D). The two-dimensional version has been previously discussed. The three-dimension version takes the 2d planes and installs them in the 3d "theater".

Major Strength: builds on previous work

Major Weirdness (current): you need to establish planes to do anything.

Major Limitation (current); no version of hidden line "assist"



<pre>loadMacros(   "PGstandard.pl",   "MathObjects.pl",   "PGchoicemacros.pl",   "Sketch3D.pl", );</pre>	<p>Sketch3D.pl should be placed in your "macros" directory.</p> <p>It is then loaded in the same manner as other packages.</p>
<pre>%ruler = ( start =&gt;"SS0W4W R l4W0W0O3O R4W4V R3W3V R2W2V R1W1V R0W0Q C0S", continue =&gt;"SS4W&gt;W R l&gt;W4W R5W5U R6W6V R7W7V R8W8V R9W9V R:W:T R;W;V R&lt;W&lt;V R=W=V R&gt;W&gt;V R3O=O C:S", );</pre>	<p>This is the text strings that make the ruler.</p> <p>The logic of this is described in the documentation for drawing 2d sketches. (uploaded previously)</p>
<pre>\$measure =Sketch3D-&gt;new(1600,0.1);</pre>	<p>No surprises here: the object is created in an ordinary way.</p> <p>The dots (1600) is set high as I am going to print these for the students.</p>
<pre>\$measure -&gt;SetObserver(2.5*\$X,-2*\$X,1.5*\$Z,0,0,0);</pre>	<p>Six arguments here:</p> <p>The first three are the x,y,z (theater) coordinates for the location of the observer.</p> <p>The second three are the x,y,z (theater) coordinates for the point in space at which the observer is looking.</p>
<pre>#\$measure -&gt;SetNoseSpin(30);</pre>	<p>For completeness only:</p> <p>The observer is looking at a point in space - he may also rotate his head. In degrees.</p>

\$measure ->MakeMatrix();

Just really needs to be there.

And as there is typically only one observer it only needs to be there once, for that observer.

The transformation is:

- translation of coordinates so that the observer is at the origin
- rotation so that the z'-axis points in the same direction that the observer is looking.
- basic triangle trigonometry.

MakeMatrix() constructs the rotation matrix required.

\$measure ->SelectPlane(0);

There is always a current plane and it is identified by a number.

\$measure ->BuildPlane(0,0,0,1,0,0,0,1,0);

Here we select a plane (ie. pick a number), and then we build it.

9 - arguments

first three - the location in theater coordinates that correspond to (0,0) in the 2d plane

second three - in theater coordinate, the direction in which the horizontal (x) 2d coordinate should point. (bases vector)

third three - in theater coordinate, the direction in which the vertical (y) 2d coordinate should point. (bases vector)

vectors are normalized - skewed vectors are at the programmer's peril.

<pre>\$measure -&gt;BuildCaptionPlane(2*\$X,2.5*\$X,,0,0,0,1);</pre>	<p>"An out-of-body experience"</p> <p>This pops and pushes to the caption stack (see the 2d documentation)</p> <p>A new plane is created with its origin at the point (3d point in theater coordinates) that was stored for a caption. A new caption is pushed to the caption stack with 2d coordinates (0,0)</p> <p>The six arguments are the bases vectors for the horizontal and vertical of the caption</p>
<pre>\$measure-&gt;caption("BRASS OR COPPER ?",1,0);</pre>	<p>The subsequent "caption" call will pop the (0,0) off the stack and the caption will be drawn at the origin of the plane that has been build.</p> <p>It was done this way as there can be other kinds of captions. I have somewhere (not here) a caption type that will draw mixed numbers. Other types of captions may well be constructed in the future.</p>
<pre>\$measure -&gt;SelectPlane(1); \$measure -&gt;BuildPlane(0,0,\$Z,1,0,0,0,1,0);  \$measure -&gt;SelectPlane(2); \$measure -&gt;BuildPlane(\$X,0,0,0,1,0,0,0,1);</pre>	<p>Two planes have been built as described above.</p> <p>the "figure"</p> <p>This has been described in the 2d documentation. except for the space SR and the space SS</p>
<pre>\$measure-&gt;figure("SSRRRR Ixyz SR RRRxRxy SSxyxRRRRyxy F BRRxy RRR SRRR RxR SSxR ST FSS C BRRyz SW FSS",(\$X,\$Y, \$Z));</pre>	<p>Here we switch planes on the fly possibly drawing a straight line between the two.</p> <p>space SR means you switch to plane R-R = 0  space SS means you switch to plane S-R = 1  space SW means you switch to plane W-R = 5</p>
<p>"Divide by Zero" errors</p>	<p>Think about it as being poked in the eye; it is the programmer's responsibility to keep the objects out of the plane of the observer</p>

