hip2wrl: A Java Program to Represent a Hipparcos Star Collection as a X3D File

Richard J. Mathar[*]
Max-Planck Institute for Astronomy, Königstuhl 17, 69117 Heidelberg, Germany
(Dated: March 26, 2024)

A Java program is presented which extracts star positions from the Hipparcos main catalogue and places them into a sphere collection rendered in a X3D, VRML97 or HTML file. The main options to the executable are a cut-off distance to some center of the scene (the sun by default) and labeling some or all of the spheres with common names, Henry Draper numbers or Hipparcos ID's.

PACS numbers: 97.10.Vm, 98.35.Pr, 01.50.F-, 97.10.Yp

I. RENDERING THE HIPPARCOS CATALOGUE

A. Choice of Output Format

The star catalogue extracted from the Hipparcos satellite mission contains 118,322 stars in the solar neighborhood [1]. As an aid to visualization of the positions we present a program that reads lines of the ASCII catalogue, filters the objects up to some maximum distance to a reference position, and puts them into an ASCII file in the Extensible 3D (X3D) format [2], the format of the older Virtual Reality Modeling Language (VRML) [3–5], or the Hypertext Markup Language (HTML). The benefit of this work is that the astrometric (that is, 3D) information fostered by the catalogue is copied into a format which allows interactive visualization and modification (rotation, translation, . . .) through standard mouse actions.

The result one may expect from installing and compiling the program and the catalogue is illustrated in Figure 1. Whether one can measure distances on the screen, display coordinate systems, and/or actually see the labels with the star names depends on the viewer in use.

B. Parameter Selection

This scanner of the Hipparcos data lines uses only a small subset of the information:

- The integer ID (H1) for various name cross referencing with other data bases defined in some tables of volume 13 of the catalogue.

- Magnitude in the visible (H5). A rough impression of (apparent) magnitude is placed on the spheres in the 3D file by dividing each component of the RGB (red, green, blue) color derived from the spectral type through $m + 1.54$. (The constant 1.54 is derived from the brightest magnitude of -1.44 in the catalogue.) This defines the *emissive* color of the spheres as the *diffuse* colors scaled by this function of magnitude.

  No attempt is made to scale magnitudes by some power laws of distance to indicate absolute magnitudes.

- The declination (H9), right ascension (H8) and parallax (H11) as a system of two angles and one distance to the origin of the celestial reference frame that are converted to three Cartesian coordinates with the standard formulae of spherical coordinates. Lines with negative parallax are skipped for all further processing.

- The spectral classification (H76). After some crude steps of name cleansing (taking only parts before a slash, ignoring dots and portions after colons) the string is used in a lookup table of RGB colors which define the diffuse color of each sphere. If that lookup fails because the cleansing does not converge to one of the tabulated colors (Appendix A 15), the sphere is shown in grey.

[*] https://www.mpia-hd.mpg.de/~mathar

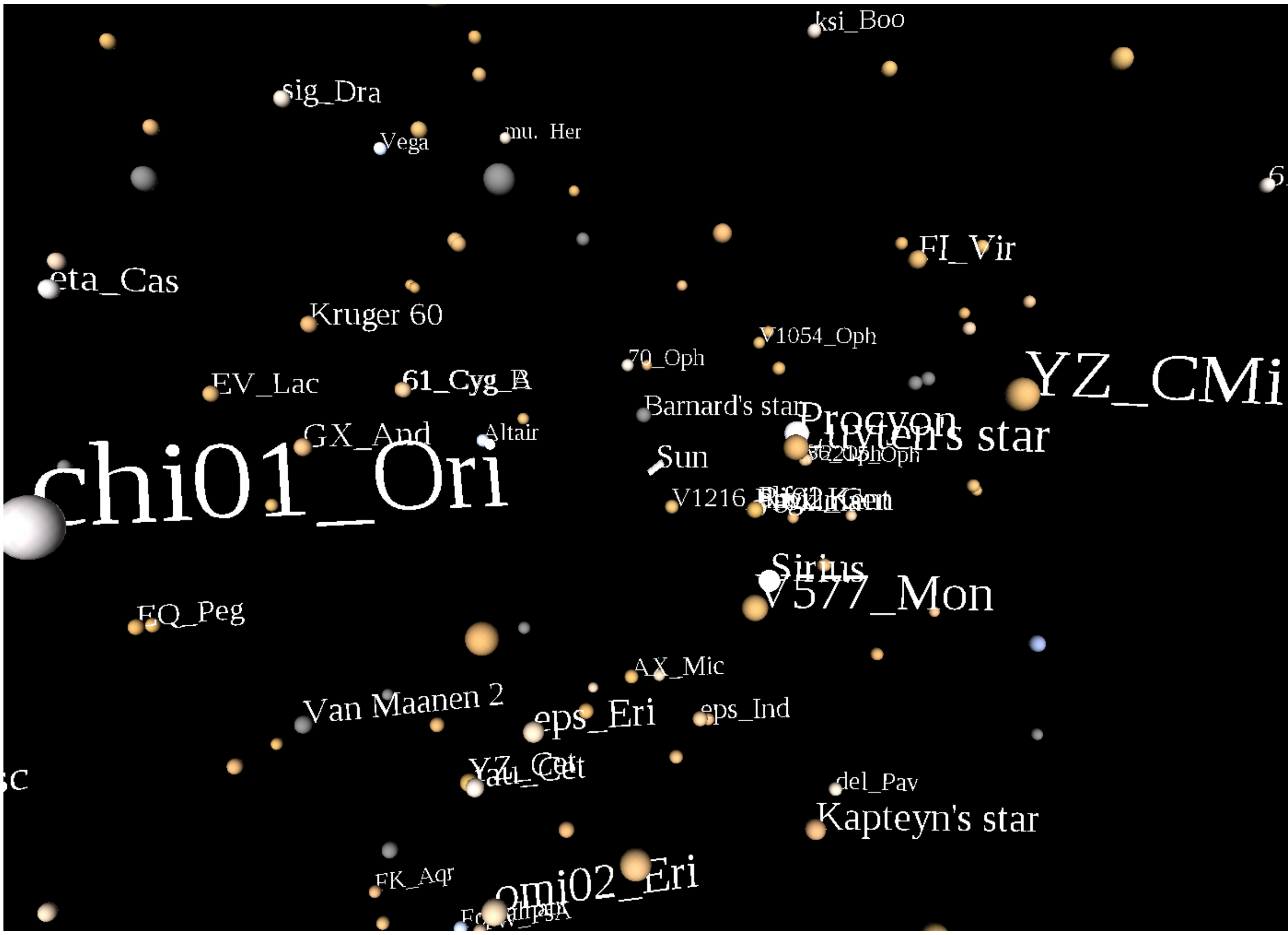

FIG. 1. Screen shot looking at a file generated with a cut-off radius of 30 light years around the sun, as presented by `freewrl` [6].

II. INVOCATION

Once the program and the auxiliary data files are compiled according to Appendix A, the program is run with a command line of the form

`java -cp . de.mpg.mpia.hip.hip2wrl` *[*`-w|-h`*] [*`-f` *hip_main.dat] [*`-l` *lightyears] [*`-o` *outfile.wrl] [*`-t` *0or1or2] [*`-C` *hipnumber] [*`-F` *0or1or2] [*`-r` *sphereradius]*

The brackets are not part of the command line but indicate that each of the 8 options may be omitted. The options define the following parameters:

- `-w, -h`: By default the program generates a X3D file. If the option `-w` is used, a VRLM97 file is generated; if the option `-h` is used, a HTML file is generated. In any case only one of the three file types is generated; the program must be called multiple times if all three formats are desired. The disadvantage of the HTML is that it needs an online browser because it calls a JavaScript bundle within `www.x3dom.org`; the HTML output is a simple wrapper around the X3D representation.

- `-f` Indicates with its argument where the main catalogue has been put into the local computer's file system during the installation phase. The default of the argument is `hip_main.dat`, which means it assumes that the original file name has been kept and is in the working directory of the user when the program is called.

- `-l` Specifies the maximum distance of stars away from the origin of the reference system in units of light years. If not specified, a value of `27.0` is used, which means of the order of 100 stars will typically be put into the output file. The run time of the program and the output file size grow roughly with the third power of the argument. The maximum distance between stars and the sun in the catalogue is ≈ 326000.

- `-o` Specifies the file name of the output file that will be created. If the option is not used, the name `hip_main.*` with suffix `.x3d` or `.wrl` or `.html` will be used.

- `-t` This option is followed by either the number `0` or the number `1` or the number `2`. If the option is not used, a default of `1` is used. Larger numbers mean that more stars are tagged with star names according to the following scheme:

  - If the number is `0`, no name tags are emitted.
  - If the number is `1`, the names of the 96 ‘common’ stars of `ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/t` of volume 13 of the catalogue are used (see the list in Appendix A 3). If the Hipparcos star does not fall into that category, the Bayer/Flamsteed name is looked up (Appendix A 14). If the Hipparcos star is not in that list either, no tag is emitted. An example is given in Figure 1 where some stars have names and others do not.
  - If the number is `2`, names are preferentially searched as with option `-t 1`. If that lookup does not succeed, HD numbers in the format `HDxxxx` are shown (Section A 13), and if there is no such entry either, the Hipparcos ID in the format `HIPxxxx` is the final name. With this choice every star in the output file is labeled.

  Note that 3D viewers like `meshlab` may not show text strings at all, even if they are in the file.

- `-C` This option puts the origin of the coordinate system at a star which is identified by its Hipparcos number. The number should be an integer in the range 1 to 118322. If the option is not used, the coordinate system is centered at the sun’s position. There are two side effects of the parameter: (i) The stars that are extracted from the catalogue are measured by their distance relative to that point of origin and included depending on the value of the `-l` option. (ii) The star at that position is shown not as a sphere but as a small brick with the longest and middle edge pointing along the plane of the ecliptic, and the shortest edge toward the North pole. To put Vega into the center of the scene use `-C 91262`, for example.

- `-F` This option is followed by either a `0`, a `1` or a `2` effecting the orientation (facing) of the name tags in the scene. The default is `0`.

  - The value of `0` uses the ‘billboard’ option of the VRML/X3D specification, which means that the texts are rotating to stay readable while the user changes the camera position. (This option was used in Figure 1.)
  - The value of `1` causes the text to stay fixed relative to the ecliptic, all baselines pointing into the $\alpha = \delta = 0$ direction as if the scribbling paper was fixed in the ecliptic. So everything is best readable when looking edge-on onto the ecliptic.
  - The value of `2` causes all text to stay fixed relative to the ecliptic. The labels are placed tangential to celestial spheres centered at the origin (which depends on the `-C` selection).

- `-r` This option sets the size of each sphere that represents the stars. If not used a default of `0.3` is inserted. Note that this is effectively a number in units of light years and is unrelated to any actual size of the objects; this number is just a mean to improve the visibility and indicating to the eye (through variable projected ball radii) which stars are foreground and which are background.

The example creating Figure 1 is

```
java -cp . de.mpg.mpia.hip.hip2wrl -l 30 -o hip30_1.wrl -r 0.3
freewrl hip30_1.wrl
```

Appendix A: Source Code

The full set of files that construct our 3D files based on a subset of stars picked from the catalogue consists of the ASCII file of the catalogue, the JAVA source code and its compiled code, and some auxiliary files that translate star numbers of the catalogue into more common name formats. Most of these files are listed in the further sections by their positions in the subdirectory structure. The code is available under the LGPL-v3 license.

The main catalogue can be downloaded from `ftp://cdsarc.u-strasbg.fr/pub/cats/I%2F239/` and should be decompressed with `gunzip` such that the file `hip_main.dat` is somewhere in the local file system. Once the file `*.java` are placed in the subdirectories, they are compiled by moving to the top directory and entering

```
javac -cp . hipparcos/tools/*.java
javac -cp . de/mpg/mpia/hip/*.java
```

The construction of the auxiliary lookup files is described individually for the files further down.

1. File Makefile

```
all: java_hipparcos hip2wrl

# compile the hipparcos subdirectory
java_hipparcos:
    cd hipparcos/tools ; make
    javac -sourcepath . hipparcos/tools/*.java

# compile the MPIA subdirectory
hip2wrl:
    cd de/mpg/mpia/hip ; make
    javac -sourcepath . de/mpg/mpia/hip/*.java

# generate docs/index.html
doc:
    javadoc -quiet -sourcepath . -d docs -private de/mpg/mpia/hip/*.java hipparcos/tools/*.java

# run some examples/tests to generate VRLM97 files
# alternative call java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 30 -t 1 -o hip30_1.wrl ;
# test: all
#     java -cp . de.mpg.mpia.hip.hip2wrl -w -f hip_main.dat -l 30 -t 0 -o hip30_0.wrl
#     java -cp . de.mpg.mpia.hip.hip2wrl -w -f hip_main.dat -l 30 -t 1 -o hip30_1.wrl
#     java -cp . de.mpg.mpia.hip.hip2wrl -w -f hip_main.dat -l 30 -t 2 -o hip30_2.wrl
#     java -cp . de.mpg.mpia.hip.hip2wrl -w -f hip_main.dat -l 50 -t 0 -o hip50_0.wrl
#     java -cp . de.mpg.mpia.hip.hip2wrl -w -f hip_main.dat -l 50 -t 1 -o hip50_1.wrl
#     java -cp . de.mpg.mpia.hip.hip2wrl -w -f hip_main.dat -l 50 -t 2 -o hip50_2.wrl
#     java -cp . de.mpg.mpia.hip.hip2wrl -w -f hip_main.dat -l 100 -t 0 -o hip100_0.wrl
#     java -cp . de.mpg.mpia.hip.hip2wrl -w -f hip_main.dat -l 100 -t 1 -o hip100_1.wrl
#     java -cp . de.mpg.mpia.hip.hip2wrl -w  -f hip_main.dat -l 100 -t 2 -o hip100_2.wrl

# run some examples/tests to generate X3D files
test: all
    java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 30 -t 0 -o hip30_0.x3d
    java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 30 -t 1 -o hip30_1.x3d
    java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 30 -t 2 -o hip30_2.x3d
    java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 50 -t 0 -o hip50_0.x3d
    java -cp . de.mpg.mpia.hip.hip2wrl -h -f hip_main.dat -l 50 -t 0 -o hip50_0.html
    java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 50 -t 1 -o hip50_1.x3d
    java -cp . de.mpg.mpia.hip.hip2wrl -h -f hip_main.dat -l 50 -t 1 -o hip50_1.html
    java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 50 -t 2 -o hip50_2.x3d
    java -cp . de.mpg.mpia.hip.hip2wrl -h hip_main.dat -l 50 -t 2 -o hip50_2.html
    java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 100 -t 0 -o hip100_0.x3d
    java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 100 -t 1 -o hip100_1.x3d
    java -cp . de.mpg.mpia.hip.hip2wrl -f hip_main.dat -l 100 -t 2 -o hip100_2.x3d

# generate a jar file
hip2wrl.jar: java_hipparcos hip2wrl doc
    jar cfe $@ de.mpg.mpia.hip.hip2wrl Makefile README de/mpg/mpia/hip/*.java docs hipparcos/tools/*.java

clean:
    - rm -rf docs
    - gzip -9 hip_main.dat
    - rm -f *.wrl
    - rm de/mpg/mpia/hip/*.class hipparcos/tools/*.class
```

2. File hipparcos/tools/DelimitedLine.java

```
package hipparcos.tools;

import java.io.*;
import java.lang.*;
import java.util.*;

/** Wraps up a line in s Stringtokenizeer using a given delimiter
provide usefull casting methods to get back doubles etc.
*/
public class DelimitedLine {
    protected StringTokenizer line;

    /** Ctor
    * @param line the string to be parsed as alime
    * @param delim the delimiter between the fields in th eline
    */
    public DelimitedLine(String line, char delim) {
        this.line = new StringTokenizer(line,new String(delim+""));
    }

    /** Get the next token in the sequence of fields.
    * @return The string in the next entry that was extracted.
    * Leading or trailing white space is removed.
    * @throws Exception If the format in the data input stream did not match the expectation.
    */
    protected String getNext() throws Exception {
        return (line.nextToken().trim());
    }

    /** Get the next token in the sequence of fields
    * @return Get the next field between the delimiters.
    * @throws Exception If the format in the data input stream did not match the expectation.
    */
    public String getNextString() throws Exception {
        return (getNext());
    }

    /** Interpret the next token as an integer
    * @return The integer that was extracted.
    * @throws Exception If the format in the data input stream did not match the expectation.
    */
    public int getNextInt() throws Exception {
        return (new Integer(getNext()).intValue());
    }

    /** Interpret the next token as a double precision number.
    * @return The doubles that was extracted.
    * @throws Exception If the format in the data input stream did not match the expectation.
    */
    public double  getNextDouble() throws Exception {
        String tmp = getNext();
        if (tmp.length() == 0) {
            throw (new Exception ());
        }
        return (new Double(tmp).doubleValue());
    }

    /** parse the string as a set of doubles and return it in an array
    * @return The sequence of doubles that were parsed.
    * @throws Exception If the format in the data input stream did not match the expectation.
    */
    public double[] getDoubleArray()  throws Exception {
        if (line.countTokens() == 0 )
            throw new Exception("Empty String for array");
        double[] dnums = new double[line.countTokens()];
        for (int t=0; t< dnums.length; t++ )  {
            dnums[t] = getNextDouble();
        }
        return dnums;
    }

    /** parse the string as a set of ints and return it in an array
    * @return The sequence of integers that were parsed.
    * @throws Exception If the format in the data input stream did not match the expectation.
    */
    public int[] getIntArray()  throws Exception {
        if (line.countTokens() == 0 )
            throw new Exception("Empty String for array");
        int[] inums = new int[line.countTokens()];
        for (int t=0; t< inums.length; t++ )  {
            inums[t] = getNextInt();
        }
        return inums;
    }

    /** Skip/ignore 1 or more fields
    * @param n The positive number (number of fields to skip)
    * @throws Exception If the format in the data input stream did not match the expectation.
    * @since 2015-07-17
    * @author R. J. Mathar
    */
    public void skip(int n)  throws Exception {
        for(int s=0 ; s < n ; s++)
        {
            getNext() ;
        }
    }
}
```

3. File hipparcos/tools/Star.java

```
package hipparcos.tools;

import java.text.*;
import java.awt.*;
import java.net.*;
import java.io.* ;
import de.mpg.mpia.hip.Point3D ;

/** A class to represent one entry in the hipparcos catalog.
* @author William O'Mullane for the Astrophysics Division of ESTEC  - part of the European Space Agency.
*/
public class Star {

    public String type; // H0 or T0 contains H or T
    public String id; // H1 ot T1
```

```
/** magnituted H5
*/
public double  mag;

/** alpha in degrees H8
*/
public double alpha;

/** delta in degrees H9
*/
public double  delta;

/** paralax in mas, H11
*/
public double  paralax;

/** proper motion mu(alpha)*cos(delta) H12
*/
public double muAlpha;
/** proper motion mu(delta) H13
*/
public double muDelta;
public double b_v; //H37
public String sptype; // spectral type
public boolean inHIPnTYC=false;
public boolean inHIP=false;

/** Default constructor.
* Sets the id (field H1) to null.
*/
public Star () {
    id=null;
}

/** Contruct the star from the delimetd line as it appears in the catalog
* @param str The ASCII line of the main catalogue
* @throws Exception if the list of strings, integers and doubles in str does not match the main catalogue format.
*/
public Star (String str) throws Exception {
    DelimitedLine line= new DelimitedLine(str,'|');
    String skip;
    type = line.getNextString();
    id   = line.getNextString();
    line.skip(3) ;
    mag  = line.getNextDouble();    //H5
    line.skip(2) ;
    alpha = line.getNextDouble();
    delta = line.getNextDouble();
    line.skip(1) ;
    paralax=line.getNextDouble();   //H11
    try {
        muAlpha = line.getNextDouble();
    } catch (Exception e)
    {};

    try {
        muDelta = line.getNextDouble(); //h13
    } catch (Exception e)
    {};

    int i=13;
    if (type.startsWith("T")) {// get T31
        /* this not for the main catalogue but the Tycho
        */
        try {
            for (i=13; i<30; i++) {
                try {
                    skip=line.getNextString();
                } catch (Exception e) {
                    System.out.println ("Skip Failed "+ e);
                }
            }
            i++;
            String tmpStr=line.getNextString(); //extractHipNo from T31
            int tmp= new Integer(tmpStr).intValue();
            inHIPnTYC=(tmp >=1);
        } catch (Exception e) {
            inHIPnTYC=false;
        }
    } else { inHIP=true; };
    while (i <  36) {
        try {
            line.skip(1) ;
            i++;
        } catch (Exception e) {
            System.out.println ("Skip Failed "+ e);
        }
    }
    b_v = line.getNextDouble(); // H37

    /* skip H38 to H75 inclusive
    */
    line.skip(38) ;
    sptype = line.getNextString(); // H76

}

/**
* @return The distance to the reference frame (light years)
* @since 2015-07-18
* @author R. J. Mathar
*/
public double ly()
{
    /* tan (parallax) = au/distance
    * distance = au/tan(parallax)
    * distance/ly = (au/ly)/tan(parallax)
    * Convert mas to as to degrees do radians
    */
    final double pax = paralax/1000.0/3600.0/Constants.degc ;
    if ( pax == 0.0)
        return 0.0 ;
    else
        return 1.0/(Math.tan(pax)*Constants.lyau) ;
}

/**
* Distance to a referencd point.
* @param ref The reference point.
* @return The distance in light years.
* @since 2015-08-09
* @author R. J. Mathar
*/
public double ly(final Point3D ref)
{
    return posit().distance(ref) ;
}
```

```
/**
* Construct the Cartesian coordinates where the ICRS ecliptic is the (x,y) plane.
* @return The three components of the position in units of light years.
* @since 2015-07-18
* @author R. J. Mathar
*/
public Point3D posit()
{
    final double r = ly() ;
    /* convert alpha and delta on the fly from degrees to radians
    */
    final double x = r*Math.cos(alpha/Constants.degc)*Math.cos(delta/Constants.degc) ;
    final double y = r*Math.sin(alpha/Constants.degc)*Math.cos(delta/Constants.degc) ;
    final double z = r*Math.sin(delta/Constants.degc) ;
    return new Point3D(x,y,z) ;
}

/**
* @param scaleMag if true scale the RGB color with some invers of the magnitude.
* @return an integer triple of 0.0..1.0 RGB colors
*/
public float[] rgbCol(final boolean scaleMag)
{
    float[] rgb = new float[3] ;
    /* default (if not found..
    */
    rgb[0]=rgb[1] =rgb[2] = 0.0f ;

    String cleanSptype = sptype;

    /* take what is after a slash ...
    int idx = cleanSptype.indexOf("/") ;
    if ( idx >= 0)
        cleanSptype = cleanSptype.substring(idx+1) ;
    */
    /* take what is before a slash ...
    */
    int idx = cleanSptype.indexOf("/") ;
    if ( idx >= 0)
        cleanSptype = cleanSptype.substring(0,idx) ;

    /* replace 1.5, 2.5, 3.5 etc by straight integers
    */
    idx = cleanSptype.indexOf(".5") ;
    if ( idx >= 0)
        cleanSptype = cleanSptype.substring(0,idx) + cleanSptype.substring(idx+2) ;

    /* delete anything after a colon
    */
    idx = cleanSptype.indexOf(":") ;
    if ( idx >= 0)
        cleanSptype = cleanSptype.substring(0,idx) ;

    /* delete anything after a triple dots
    */
    idx = cleanSptype.indexOf("...") ;
    if ( idx >= 0)
        cleanSptype = cleanSptype.substring(0,idx) ;

    /* delete anything after a plus
    */
    idx = cleanSptype.indexOf("+") ;
    if ( idx >= 0)
        cleanSptype = cleanSptype.substring(0,idx) ;
    /* simple types like G0 or A2... converted to G0III etc
    */
    String extSptype =cleanSptype;
    if ( cleanSptype.length() == 2 )
    {
        if ( sptype.startsWith("D") )
            extSptype = cleanSptype + "3" ;
        else
            extSptype = cleanSptype + "III" ;
    }

    /* http://www.isthe.com/chongo/tech/astro/HR-temp-mass-table-byhrclass.html
    */
    URL rgbtxt = getClass().getResource("StarRGB.txt") ;
    try
    {
        FileReader istream = new FileReader(rgbtxt.getFile()) ;
        BufferedReader dstream = new BufferedReader(istream) ;
        for(;;)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                break ;
            /* split at tab */
            String[] linFields = lin.split("\t") ;
            if ( extSptype.toUpperCase().startsWith(linFields[0]) )
            {
                for( int c=0 ; c < 3 ; c++)
                {
                    /* rgb values are originaly in the range 0..255.
                    * Convert them to VRML units by division through 256.
                    */
                    rgb[c] = (new Short(linFields[1].substring(4*c,4*c+3))).floatValue()/256.0f ;
                }
                break;
            }

        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
    }

    if ( rgb[0] == 0.0f && rgb[1] == 0.0f && rgb[2] == 0.0f)
    {
        rgb[0] = rgb[1] = rgb[2] = 0.6f ;
    }

    if ( scaleMag)
        for( int c=0 ; c < 3 ; c++)
        {
            /* Star magnitudes are in the range >= -1.44, so we take this
            * as the "origin" of the luminosity: mag+1.44.
            */
            rgb[c] /= 1.54+mag ;
        }

    return rgb ;
} /* rgbCol */

/**
```

```
* @return The magnitude H5
*/
public double getMag() {
    return mag;
}

/**
* @return The alpha coordinate (deg) H8
*/
public double getAlpha() {
    return alpha;
}

/**
* @return The delta coordinate (deg) H9
*/
public double getDelta() {
    return delta;
}

/**
* @return The astrometric paralax (mas) H11
*/
public double getParalax() {
    return paralax;
}


/**
* @return Hipparcos or Tycho id (as a string)
*/
public String getId() {
    return id;
}

/**
* @return First field in the catalogue. "H" or "T".
*/
public String getType() {
    return type;
}

/**
* @return True if the star is in the Hipparcos catalogue. False if in the Tycho catalogue.
*/
public boolean inHIP() {
    return inHIP;
}

/**
* @param years The number of years extrapolated from the epoch.
* @return The total change in alpha due to proper motion (deg)
*/
public double getMuAlpha(int years) {
    return (years*(muAlpha/3600000));
}


/**
* Translate a HIP number to a Bayer/Flamsteed star name of volume 13 of the main catalogue
* @return A more common name. This is empty if there is none.
* @since 2015-07-27
*/
public String hip2ident4()
```

```
{
    final int hipid = (new Integer(id).intValue());
    return hip2ident4(hipid) ;
}

/**
* Translate a HIP number to a Variable star name of volume 13 of the main catalogue
* @return A name of table ID5. This is empty if there is none.
* @since 2015-07-29
*/
public String hip2ident5()
{
    final int hipid = (new Integer(id).intValue());
    return hip2ident5(hipid) ;
}

/**
* Translate a HIP number to a Henry-Draper (HD) number name of volume 13 of the main catalogue
* @return A name of table ID2. This is empty if there is none.
* @since 2015-07-29
*/
public int hip2ident2()
{
    final int hipid = (new Integer(id).intValue());
    return hip2ident2(hipid) ;
}

/**
* Translate a HIP number to a star name of volume 13 of the main catalogue
* @param hipid The index of the star in the main catalogue.
* @return A more common name refering to the zodiac sign.
*   This is empty if there is none (as there are 4440 assignments listed, which
*   is only a minor portion of roughly 4 percent of the entire catalogue)
* @since 2015-07-27
*/
static public String hip2ident4(int hipid)
{
    /* ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident4.doc.gz
    * dos2unix < ident4.doc | tr "|" " " | awk '{print $2,$1}' | sort -u -k 1n > ident4.txt
    */
    URL idtbl = hipparcos.tools.Star.class.getResource("ident4.txt") ;
    try
    {
        FileReader istream = new FileReader(idtbl.getFile()) ;
        BufferedReader dstream = new BufferedReader(istream) ;
        for(;;)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                break ;
            /* split at first blank */
            String[] linFields = lin.split(" ") ;
            final int namid = (new Integer(linFields[0]).intValue());
            if ( namid == hipid)
                return linFields[1] ;
            else if ( namid > hipid)
                /* because ident4.txt is sorted numerically in the
                * first field, we can skip the rest because thy will not match either
                */
                break;

        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
    }
    return new String() ;
}

/**
* Translate a HIP number to a star name of volume 13 of the main catalogue
* @param hipid The index of the star in the main catalogue.
* @return A more common name refering to the variable star names.
*   This is empty if not covered by table ID5.
* @since 2015-07-29
*/
static public String hip2ident5(int hipid)
{
    /* ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident5.doc.gz
    * dos2unix < ident5.doc | tr "|" " " | awk '{print $2,$1}' | sort -n > ident5.txt
    */
    URL idtbl = hipparcos.tools.Star.class.getResource("ident5.txt") ;
    try
    {
        FileReader istream = new FileReader(idtbl.getFile()) ;
        BufferedReader dstream = new BufferedReader(istream) ;
        for(;;)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                break ;
            /* split at first blank */
            String[] linFields = lin.split(" ") ;
            final int namid = (new Integer(linFields[0]).intValue());
            if ( namid == hipid)
                return linFields[1] ;
            else if ( namid > hipid)
                /* because ident5.txt is sorted numerically in the
                * first field, we can skip the rest because thy will not match either
                */
                break;

        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
    }
    return new String() ;
}

/**
* Translate a HIP number to a HD number of volume 13 of the main catalogue
* @param hipid The index of the star in the main catalogue.
* @return The Henry Draper number according to the ID2 table.
*   This is 0 if not covered by table ID2.
* @since 2015-07-29
*/
static public int hip2ident2(int hipid)
{
    /* ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident2.doc.gz
    * dos2unix < ident2.doc | tr "|" " " | awk '{print $2,$1}' | sort -n > ident2.txt
    */
    URL idtbl = hipparcos.tools.Star.class.getResource("ident2.txt") ;
    try
    {
        FileReader istream = new FileReader(idtbl.getFile()) ;
        BufferedReader dstream = new BufferedReader(istream) ;
        for(;;)
        {
            String lin = dstream.readLine() ;
            if ( lin == null)
                break ;
            /* split at first blank */
            String[] linFields = lin.split(" ") ;
            final int namid = (new Integer(linFields[0]).intValue());
            if ( namid == hipid)
                return (new Integer(linFields[1])).intValue() ;
            else if ( namid > hipid)
                /* because ident2.txt is sorted numerically in the
                * first field, we can skip the rest because thy will not match either
                */
                break;

        }
    }
    catch (Exception ex)
    {
        System.err.println(ex) ;
        ex.printStackTrace() ;
    }
    return 0 ;
}

/**
* Translate a HIP number to a common star name
* @return A more common name.
*  If none is found, the HIPxxxx string with just the hip is returned.
*/
public String hip2name()
{
    final int hipid = (new Integer(id).intValue());
    return hip2name(hipid) ;
}

/**
* Translate a HIP number to a common star name.
* Equivalent to an annex in vol 13 of the catalogue.
* @param hip The star number in the main catalogue
* @return A more common name.
*  If none is found, the HIPxxxx string with just the hip is returned.
*/
static public String hip2name(int hip)
{
    /* http://www.rssd.esa.int/index.php?project=HIPPARCOS&page=common
    * ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident6.doc.gz
    */
    final String[] ident6 = {
    "Alpheratz",
    "Caph",
    "Algenib",
    "Ankaa",
    "Shedir",
    "Diphda",
    "Van Maanen 2",
    "Mirach",
    "Achernar",
    "Almaak",
    "Hamal",
    "Mira",
    "Polaris",
    "Acamar",
    "Menkar",
    "Algol",
    "Mirphak",
    "Alcyone",
    "Pleione",
    "Zaurak",
    "Aldebaran",
    "Kapteyn’s star",
    "Rigel",
    "Capella",
    "Bellatrix",
    "Alnath",
    "Nihal",
    "Mintaka",
    "Arneb",
    "Alnilam",
    "Alnitak",
    "Saiph",
    "Betelgeuse",
    "Red Rectangle",
    "Canopus",
    "Alhena",
    "Sirius",
    "Adhara",
    "Luyten’s star",
    "Castor",
    "Procyon",
    "Pollux",
    "Alphard",
    "Regulus",
    "Algieba",
    "Merak",
    "Dubhe",
    "Denebola",
    "Groombridge 1830",
    "Phad",
    "Megrez",
    "Acrux",
    "3C 273",
    "Alioth",
    "Cor Caroli",
    "Vindemiatrix",
    "Mizar",
    "Spica",
    "Alcor",
    "Alkaid",
    "Agena",
    "Hadar",
    "Thuban",
    "Arcturus",
    "Proxima",
    "Rigil Kent",
    "Izar",
    "Kocab",
    "Alphekka",
    "Unukalhai",
    "Antares",
    "Rasalgethi",
    "Shaula",
    "Rasalhague",
    "Etamin",
    "Barnard's star",
    "Kaus Australis",
    "Vega",
    "Sheliak",
    "Nunki",
    "Albireo",
    "Campbell's star",
    "Tarazed",
    "Altair",
    "Alshain",
    "Cyg X-1",
    "Deneb",
    "Alderamin",
    "Enif",
    "Sadalmelik",
    "Alnair",
    "Kruger 60",
    "Babcock's star",
    "Fomalhaut",
    "Scheat",
    "Markab"
    } ;

    final int[] hips ={
    677,
    746,
    1067,
    2081,
    3179,
    3419,
    3829,
    5447,
    7588,
    9640,
    9884,
    10826,
    11767,
    13847,
    14135,
    14576,
    15863,
    17702,
    17851,
    18543,
    21421,
    24186,
    24436,
    24608,
    25336,
    25428,
    25606,
    25930,
    25985,
    26311,
    26727,
    27366,
    27989,
    30089,
    30438,
    31681,
    32349,
    33579,
    36208,
    36850,
    37279,
    37826,
    46390,
    49669,
    50583,
    53910,
    54061,
    57632,
    57939,
    58001,
    59774,
    60718,
    60936,
    62956,
    63125,
    63608,
    65378,
    65474,
    65477,
    67301,
    68702,
    68702,
    68756,
    69673,
    70890,
    71683,
    72105,
    72607,
    76267,
    77070,
    80763,
    84345,
    85927,
    86032,
    87833,
    87937,
    90185,
    91262,
    92420,
    92855,
    95947,
    96295,
    97278,
    97649,
    98036,
    98298,
    102098,
    105199,
    107315,
    109074,
    109268,
    110893,
    112247,
    113368,
    113881,
    113963
    } ;
    /* first attempt: get it from ident6 assignments (common star names)
    */
    for(int j=0 ; j < hips.length ; j++)
    {
        if ( hips[j] == hip)
            return ident6[j] ;
        else if ( hips[j] > hip)
```

```
                break ;
        }

        /* second backup attempt: get it from the ident4 assignments (Bayer/Flamsteed names)
        */
        String i6 = hip2ident4(hip) ;

        if ( i6.length() > 0 )
            return i6 ;

        /* third backup attempt: get it from the ident5 assignments (VS names)
        */
        String i5 = hip2ident5(hip) ;

        if ( i5.length() > 0 )
            return i5 ;

        /* fourth backup attempt: get it from the ident2 assignments (HD numbers)
        */
        int i2 = hip2ident2(hip) ;

        if ( i2 > 0 )
            return new String("HD"+i2) ;

        /* final refuge is the HIP number itself
        */
        return new String("HIP"+hip) ;
    } /* hip2name */

} /* Star */
```

4. File hipparcos/tools/Constants.java

```
package hipparcos.tools;

/** Constants used in the tools
*/
public class Constants {
    /** Eulers constant 3.14159..
    */
    public static final double PI = Math.PI;


    /** Pi/2
    */
    public static final double cPi = 2.0*Math.atan(1);

    /** Pi/360
    */
    public static final double cPr = cPi/180;

    /** 180/Pi. Conversion factor radians to degrees
    */
    public static final double degc=57.295779513082320876798154814 ;

    /** Ratio of light year over A.U.
    * ratio 9,460,730,472,580.8 km / 149 597 870, 700 km
    * @since 2015-07-18
    */
    public static final double lyau = 63241.077084266280268653583182 ;

    public static final int vlev = 2;
} /* Constants */
```

5. File hipparcos/tools/StarFactory.java

```
package hipparcos.tools;

import java.io.*;
import java.util.*;
import java.net.URL;

/** For given alpha delta and d get all stars from the database
in that area. Optional hipOnly may be given.
* @author William O'Mullane for the Astrophysics Division of ESTEC  - part of the European Space Agency.
*/
public class StarFactory  {
    protected boolean hipOnly=false;
    protected double alpha=0;
    protected double delta=0;
    protected double d=0;
    protected BufferedReader dstream=null;
    protected boolean finished=false;
    protected boolean hipDone=false;
    protected boolean opened=false;
    protected boolean disk=false;
    protected String[] catprogs = {"shipmainra","stycmainra"};


    /**
    * @param hipOnly If true, tycho catalogues are not scanned.
    */
    protected StarFactory(boolean hipOnly) {
        this.hipOnly = hipOnly;
    } ;

    /**
    * @param alpha The alpha (RA) coordinate (degrees)
    * @param delta The delta (DEC) coordinate (degrees)
    * @param d A value for the box parameter of the URL query.
    */
    public StarFactory (double alpha, double delta, double d) {
        this.alpha = alpha;
        this.delta = delta;
        this.d = d;
        finished = ! openStream();
    }

    /**
    * @param alpha The alpha (RA) coordinate (degrees)
    * @param delta The delta (DEC) coordinate (degrees)
    * @param d A value for the box parameter of the URL query.
    * @param hipOnly If true, tycho catalogues are not scanned.
    */
    public StarFactory ( double alpha, double delta, double d, boolean hipOnly ) {
        this.hipOnly = hipOnly;
        this.alpha = alpha;
        this.delta = delta;
        this.d = d;
        finished = ! openStream();
    }

    /**
```

```
* @param fname The file name of the main catalogue in the local disks
* @since 2015-07-17
* @author R. J. Mathar
*/
public StarFactory ( String fname)
{
    finished = false ;
    opened = loadFromDisk(fname) ;
}

/**
* @return True if the input stream could be opened successfully.
*/
protected boolean openStream() {
    opened=true;
    finished=false;
    return loadFromDisk(catprogs[0]);
}

/**
* @param fname The file name with the ASCII catalog
* @return true if sucess, false if unable to open.
* @author R. J. Mathar
* @since 2015-07-17
*/
protected boolean loadFromDisk(String fname) {
    try {
        FileReader istream  = new FileReader(fname);
        dstream = new BufferedReader(istream);
    } catch (Exception e) {
        System.err.println("loadFromDisk " +fname + " "+ e);
        e.printStackTrace();
        return false;
    }
    return true;
}


/** Skip to next PRE tag in html stream.
* @throws Exception If reading line of the data input stream did not suceed.
*/
protected void skipToData() throws Exception {
    String str;
    boolean found=false;
    while (!found ) { // skip to pre
        str=dstream.readLine();
        found = (str == null) || str.startsWith("<pre>") || str.startsWith("<PRE>");
    }
    str=dstream.readLine(); // skip header
}

/** Keep getting stars until none left.
* @return The Star in the next line of the input catalogue
* @throws NoMoreStars if the catalogue has been entirely read to the end.
*/
public Star getNext() throws NoMoreStars {
    if (!opened)
        openStream();

    if (finished)
        throw new NoMoreStars();

    boolean found=false;
    String str;
```

```
        Star star=null;
        while (!found ) {
            str=null;
            try {
                str=dstream.readLine();
            } catch (Exception e) {
            };

            if (str==null) {
                if (disk && catprogs[1]!=null && !hipDone && !hipOnly) {
                    loadFromDisk(catprogs[1]);
                    hipDone=true;
                    try {
                        str=dstream.readLine();
                    } catch (Exception tyce) {
                        tyce.printStackTrace();
                        throw new NoMoreStars();
                    }
                } else {
                    finished = true;
                    try {
                        dstream.close();
                    } catch (Exception e) {};
                    throw new NoMoreStars();
                }
            }
            else
            {
                if (str.startsWith("</pre>") || str.startsWith("</PRE>")) {
                    if (hipDone || hipOnly ) {
                        finished=true;
                        try {
                            dstream.close();
                        } catch (Exception e) {};
                        throw new NoMoreStars();
                    } else {
                        try {
                            skipToData();
                            str=dstream.readLine();
                        } catch (Exception e) {
                            System.err.println("StarFactory:header "+e);
                            throw new NoMoreStars();
                        }
                        hipDone=true;
                    }
                }
            }
            try {
                star= new Star(str);
                found=true;
            } catch (Exception e) {
            }
        }
        return star;
    }
} /* StarFactory */
```

6. File hipparcos/tools/NoMoreStars.java

```
package hipparcos.tools;
```

```
/** Exception thrown when a factory has no more stars
*/
public class NoMoreStars extends Exception {
    public NoMoreStars() {};
    public NoMoreStars(String msg) {
        super(msg);
    };
}
```

7. File de/mpg/mpia/hip/hip2wrl.java

```
package de.mpg.mpia.hip ;

import hipparcos.tools.* ;

/**
* A translator of Hipparcos stars into a VRML97 scene.
* @since 2015-07-18
* @author Richard J. Mathar
*/
class hip2wrl {

    /** Define a new center of the scene at some specific star
    * @param cHip the number in the main catalogue for the new center
    * @return The location in the global coordinate system
    */
    static Point3D locateHip(int cHip, final String catfname)
    {
        if ( cHip >0 && cHip <= 118322)
        {
            StarFactory cat = new StarFactory(catfname) ;
            for(;;)
            {
                try
                {
                    Star s = cat.getNext() ;
                    int thisid = (new Integer(s.getId())).intValue() ;
                    if ( s.ly() > 0. && thisid == cHip)
                    {
                        return  s.posit() ;
                    }
                }
                catch (Exception ex)
                {
                    break;
                }
            }
        }
        return null ;
    }

    /**
    * @param argv Vector of the command line arguments
    * @since 2024-03-24 Html output optional
    */
    static public void main(String argv[])
    {
        String catfname = new String("./hip_main.dat") ;
        String ofname = new String() ;
        /* default limit of distance to center is 28 light years
        */
        double lyLim = 28.0 ;
```

```
/* standard size of sphere radius is  0.3 (light years)
*/
double rad = 0.3 ;

int tagVerb = 1 ;
/* 0 = billboard, 1=fixed, 2=facing center
*/
int tagCntr = 0 ;

/* mid point in the scene. By default at the position of the sun.
*/
Point3D cntr = new Point3D() ;

int cHip = 0 ;

/* if true generate VRLM97 output, else x3d 3.3 or html
*/
boolean VRMLout = false ;

/* if true generate HTML output, else x3d 3.3 or VRLM97
*/
boolean Htmlout = false ;

/* collect options
*  -l <limitlightyears>
*  -f <filenamecatalog>
*  -t [0,1,2]
*  -o <outfile>
*  -F [0,1,2]
*  -r <radius/ly>
*  -C <HIPid>
*  -w
*  -h
*/
for(int i=0 ; i < argv.length; i++)
{
    if ( argv[i].compareTo("-l") == 0 )
        lyLim = (new Double(argv[i+1])).doubleValue() ;
    else if ( argv[i].compareTo("-r") == 0 )
        rad = (new Double(argv[i+1])).doubleValue() ;
    else if ( argv[i].compareTo("-f") == 0 )
        catfname = argv[i+1] ;
    else if ( argv[i].compareTo("-w") == 0 )
        VRMLout = true ;
    else if ( argv[i].compareTo("-h") == 0 )
        Htmlout = true ;
    else if ( argv[i].compareTo("-o") == 0 )
        ofname = argv[i+1] ;
    else if ( argv[i].compareTo("-t") == 0 )
        tagVerb = (new Integer(argv[i+1])).intValue() ;
    else if ( argv[i].compareTo("-F") == 0 )
        tagCntr = (new Integer(argv[i+1])).intValue() ;
    else if ( argv[i].compareTo("-C") == 0 )
    {
        cHip = (new Integer(argv[i+1])).intValue() ;
    }
}

if ( ofname.length() == 0)
{
    if  ( VRMLout)
        ofname = new String("./hip_main.wrl") ;
    else if  ( Htmlout)
```

```
        ofname = new String("./hip_main.html") ;
    else
        ofname = new String("./hip_main.x3d") ;
}

/* if the user wished to relocate the center
*/
if ( cHip > 0)
{
    Point3D c = locateHip(cHip,catfname) ;
    if ( c != null)
        cntr = c;
}

SphereSet allSphs = new SphereSet() ;

/* start with the sun
*/
Star sun = new Star() ;
if ( sun.ly(cntr) < lyLim)
{
    sun.id = new String("0") ;
    sun.paralax = 0.0 ;
    sun.type = new String("H") ;
    sun.alpha = sun.delta=0.0 ;
    sun.sptype = new String("G2V") ;

    Sphere sph0 = new Sphere(0, 0, 0,rad) ;
    sph0.rgb = sun.rgbCol(false) ;
    sph0.rgbEmit = sun.rgbCol(true) ;
    sph0.tag = new String("Sun") ;
    allSphs.add(sph0) ;
}

StarFactory cat = new StarFactory(catfname) ;
for(;;)
{
    try
    {
        Star s = cat.getNext() ;
        if ( s.ly(cntr) < lyLim & s.ly() > 0.)
        {
            /* java -cp . de.mpg.mpia.hip.hip2wrl -l 20000 | sort -u
            */

            Point3D pos = cntr.to(s.posit()) ;
            Sphere sph = new Sphere(pos.coo[0], pos.coo[1], pos.coo[2],rad) ;
            sph.rgb = s.rgbCol(false) ;
            sph.rgbEmit = s.rgbCol(true) ;
            sph.tag = s.hip2name() ;

            allSphs.add(sph) ;
        }
    }
    catch (Exception ex)
    {
        break;
    }
}


if ( VRMLout)
{
    String cmt = new String("# " + allSphs.sphs.size() + " Hipparcos objects within " + lyLim + " light years")
```

```
            allSphs.writeVrml(ofname,new String("hip_main.dat"),cmt,tagVerb,tagCntr,2.0) ;
        }
        else
        {
            String cmt = new String("<!-- " + allSphs.sphs.size() + " Hipparcos objects within " + lyLim + " light year
            if ( Htmlout)
                allSphs.writeHtml(ofname,new String("hip_main.dat"),cmt,tagVerb,tagCntr,2.0) ;
            else
                allSphs.writeX3d(ofname,new String("hip_main.dat"),cmt,tagVerb,tagCntr,2.0) ;
        }
    }

} /* hip2wrl */
```

8. File de/mpg/mpia/hip/Sphere.java

```
package de.mpg.mpia.hip ;

import java.lang.* ;
import java.util.* ;

/** A Sphere of fixed radius and center point.
* @author Richard J. Mathar
* @since 2011-07-31
*/
public class Sphere extends Point3D  {
    /** Radius
    */
    public double radius ;

    /** 3 RGB color values in the range 0 to 1
    */
    float[] rgb ;

    /** RGB color values reduced via magnitudes.
    * Used to describe emittances.
    */
    float[] rgbEmit ;

    String tag ;

    /** Ctor given the center and radius.
    * @param centr The coordinates of the center
    * @param R The radius.
    */
    public Sphere(final Point3D centr, double R)
    {
        super(centr.coo[0],centr.coo[1],centr.coo[2]) ;
        radius = R ;
        rgb = new float[3] ;
        rgbEmit = new float[3] ;
        rgb[0] =rgb[1] =rgb[2] = 0 ;
        rgbEmit[0] =rgbEmit[1] =rgbEmit[2] = 0 ;
        tag = new String() ;
    } /* Sphere */

    /** Ctor given the center and radius.
    * @param centr The coordinates of the center
    * @param R The radius.
    */
    public Sphere(final double[] centr, double R)
    {
        super(centr) ;
        radius = R ;
        rgb = new float[3] ;
        rgbEmit = new float[3] ;
        rgb[0] =rgb[1] =rgb[2] = 0 ;
        rgbEmit[0] =rgbEmit[1] =rgbEmit[2] = 0 ;
        tag = new String() ;
    } /* Sphere */

    /** Ctor given the center and radius.
    * @param centrx The Cartesian x coordinate of the center
    * @param centry The y coordinate of the center
    * @param centrz The z coordinate of the center
    * @param R The radius.
    */
    public Sphere(final double centrx, final double centry, final double centrz, double R)
    {
        super(centrx,centry,centrz) ;
        radius = R ;
        rgb = new float[3] ;
        rgbEmit = new float[3] ;
        rgb[0] =rgb[1] =rgb[2] = 0 ;
        rgbEmit[0] =rgbEmit[1] =rgbEmit[2] = 0 ;
        tag = new String() ;
    } /* Sphere */

    /**
    * @since 2011-08-02
    * @return Representation as x y z r .
    */
    public String toString()
    {
        return (super.toString() + " r " + radius );
    }

} /* Sphere */
```

9. File de/mpg/mpia/hip/Point3D.java

```
package de.mpg.mpia.hip ;

import java.util.* ;
import java.text.* ;

/** A point in 3D.
* @author Richard J. Mathar
* @since 2011-05-12
*/
public class Point3D {
    /** coo[0..2] are the Cartesian coordinates
    */
    public double[] coo ;

    /** Ctor for a point at the origin of coordinates
    * @since 2011-09-20
    */
    public Point3D()
    {
        coo = new double[3] ;
        coo[0] = coo[1] = coo[2] = 0. ;
    }
```

```
/** Ctor with the three cartesian coordinates.
* @param x The cartesian x coordinate.
* @param y The cartesian y coordinate.
* @param z The cartesian z coordinate.
*/
public Point3D(double x, double y, double z)
{
    coo = new double[3] ;
    coo[0] = x ;
    coo[1] = y ;
    coo[2] = z ;
}

/** Ctor with the three cartesian coordinates.
* @param x The cartesian x coordinate.
* @param y The cartesian y coordinate.
* @param z The cartesian z coordinate.
*/
public Point3D(final Double x, final Double y, final Double z)
{
    this(x.doubleValue(), y.doubleValue(), z.doubleValue() ) ;
}

/** Ctor with the three cartesian coordinates.
* @param x The cartesian x coordinate.
* @param y The cartesian y coordinate.
* @param z The cartesian z coordinate.
*/
public Point3D(float x, float y, float z)
{
    coo = new double[3] ;
    coo[0] = x ;
    coo[1] = y ;
    coo[2] = z ;
}

/** Ctor from a vector in 3D
* @param xyz The vector ofthe three cartesian coordinates.
*/
public Point3D(final double[] xyz)
{
    coo = new double[3] ;
    coo[0] = xyz[0] ;
    coo[1] = xyz[1] ;
    coo[2] = xyz[2] ;
}

/** Ctor from a vector in 3D
* @param xyz The vector ofthe three cartesian coordinates.
*/
public Point3D(final float[] xyz)
{
    coo = new double[3] ;
    coo[0] = xyz[0] ;
    coo[1] = xyz[1] ;
    coo[2] = xyz[2] ;
}

/** Ctor with the a tilt angle and an azimuthal direction.
* The result is a vector of length unity.
* @param z Zenith (tilt) angle or direction (rad)
* @param A Azimuth angle or direction (rad)
*/
public Point3D(double z, double A)
```

```
{
    coo = new double[3] ;
    coo[0] = Math.sin(z)*Math.cos(A) ;
    coo[1] = Math.sin(z)*Math.sin(A) ;
    coo[2] = Math.cos(z) ;
}

/** Deep copy.
*/
protected Point3D clone()
{
    return new Point3D(coo) ;
}

/** Return a copy, normalized to unit length
* @return a shrunk or stretched copy with length 1.
* @since 2011-06-15
*/
public Point3D normalize()
{
    final double l = len() ;
    return new Point3D(coo[0]/l, coo[1]/l, coo[2]/l ) ;
}

/** Difference vector to another point.
* @param oth The other point at the tail of the difference vector.
* @return diff where this + diff = oth in the vectorial sense.
*/
public Point3D to(final Point3D oth)
{
    return new Point3D( oth.coo[0]-coo[0], oth.coo[1]-coo[1], oth.coo[2]-coo[2]) ;
}

/** Difference vector to another point.
* @param oth The other point at the tail of the difference vector.
* @param t The (signed) stretch parameter for oth.
* @return this - t* oth.
*/
public Point3D subtract(final Point3D oth,final double t)
{
    return new Point3D( coo[0]-t*oth.coo[0], coo[1] -t*oth.coo[1], coo[2] - t*oth.coo[2]) ;
}

/** Difference vector to another point.
* @param oth The other point at the tail of the difference vector.
* @return this - oth.
*/
public Point3D subtract(final Point3D oth)
{
    /* somthing avoiding the implicit call (and probably faster
    */
    return new Point3D( coo[0]-oth.coo[0], coo[1] -oth.coo[1], coo[2] - oth.coo[2]) ;
}


/** Distance to another point.
* @param oth The other point to be measured in distance.
* @return The square root of the sum of squares of the Cartesian components.
* @since 2011-08-02
*/
public double distance(final Point3D oth)
{
    return subtract(oth).len() ;
}
```

```
/** Length, considering the components as a vector.
* @return The square root of the sum of squares of the Cartesian components.
*/
public double len()
{
    /* perhaps faster than
    * return Math.hypot(Math.hypot(coo[0],coo[1]),coo[2] );
    */
    return Math.sqrt(coo[0]*coo[0] + coo[1]*coo[1] + coo[2]*coo[2] );
}


/** blank separated string.
* @return Three blank-separated numbers of the 3 Cartesian coordinates.
*/
public String toString()
{
    return new String(coo[0]+ " " + coo[1] + " " + coo[2]) ;
}


/** Construct a rotation and angle that moves  the point perpendicular to the direction of its direction.
* @return A vector  with components [0,1,2] meanign the rotation axis and [3] the rotation angle (radians).
* @since 2015-07-31
*/
public double[] perp()
{

    /* Given the coo[0,1,2] with x = r cos phi sin theta, y = r sin phi sin theta and z=r cos theta,
    * (theta polar angle, phi azimuth in the spherical coordinates) we construct the
    * orthogonal matrix that rotates z = (0,0,1) into ( cos phi sin(theta-90), sin(phi) sin(theta-90),cos(theta-90)
    * = ( -cos phi cos theta, -sin(phi) cos theta, sin theta)
    * and y=(0,1,0) into
    * (cos phi sin theta, sin phi sin thea, cos theta). This is equivalent to constructing
    * a vector that is orthogonal to the original (x,y,z) vector.
    * Of course this sends the cross product x=y X z into the [cos phi cos theta, sin phi  cos theta , -sin theta ]
    * This defines the three columns of the rotation matrix.
    * (sp     cp st    -cp st)
    * (-cp    sp st    -sp st)
    * (0       ct      st)
    * Match this with the rotation
    * matrix of Appendix A in arxiv:1410.1885 by computing the direction of the rotation
    * axis (which has eigenvalue 1). The trace of the matrix is 1+2*cos(rotation angle)
    * = sin(phi)+sin(phi)*sin(theta)+sin(theta)
    */

    /* y/x = sin(phi)/cos(phi)
    */
    final double phi = Math.atan2(coo[1],coo[0]) ;
    final double cphi = Math.cos(phi) ;
    final double sphi = Math.sin(phi) ;
    /* sqrt(x^2+y^2) = r*sin(theta) >0
    */
    final double xys = Math.hypot(coo[0],coo[1]) ;
    /* sqrt(x^2+y^2)/z = sin(theta)/cos(theta)
    */
    final double theta = Math.atan2(xys,coo[2]) ;
    final double ctheta = Math.cos(theta) ;
    final double stheta = Math.sin(theta) ;

    double[] rax = new double[4] ;
    /* rotation angle (radians)
```

```
        */
        rax[3] = Math.acos( (sphi+sphi*stheta+stheta -1.0)/2.0 );

        /* sine of the rotation angle
        */
        final double srot = Math.sin(rax[3]) ;

        if ( srot == 0. || len() < 1.e-13)
        {
            /* avoid degenerate division through 0 further down
            */
            rax[0] = rax[1] =0.0 ;
            rax[2] = 1.0 ;
            rax[3] = 0.0 ;
        }
        else
        {
            /*  Omega[3,2]-Omega[2,3] = ct +sp*st = 2*sin(rot angle)*first component
            */
            rax[0] = (ctheta + sphi*stheta)/(2.*srot) ;

            /*  Omega[1,3]-Omega[3,1] = -cp*st = 2*sin(rot angle)*second component
            */
            rax[1] = -cphi*stheta/(2.*srot) ;

            /*  Omega[2,1]-Omega[1,2] = -cp-cp*st = -cp(1+st) = 2*sin(rot angle)*third component
            */
            rax[2] = -cphi*(1.0+stheta)/(2.*srot) ;
        }

        return rax ;
    } /* perp */

} /* Point3D */
```

10. File de/mpg/mpia/hip/SphereSet.java

```
package de.mpg.mpia.hip ;

import java.io.* ;
import java.util.* ;
import java.text.* ;

/** A collection of elements of type Sphere.
* @author Richard J. Mathar
* @since 2012-02-18
*/
public class SphereSet {

    /** The individual spheres, a sphere set.
    */
    public Vector<Sphere> sphs ;

    /** end-of-line separator in files
    */
    static String eol = System.getProperty("line.separator") ;

    /** Ctor for an empty set of spheres.
    */
    public SphereSet()
    {
        sphs = new Vector<Sphere>() ;
    } /* SphereSet */

    /** Ctor bundling a set of existing spheres.
    * @param spheres The vector of spheres to be collected.
    */
    public SphereSet(Vector<Sphere> spheres)
    {
        sphs = spheres ;
    } /* SphereSet */

    /** Add a sphere.
    * @param s The sphere to be added
    */
    public void add(final Sphere s)
    {
        sphs.add(s) ;
    } /* add */

    /** Create one X3D file which shows the spheres.
    * @param ofname Name of the output file
    * @param sname Name of the associated assembly/part
    * @param cmt Additional comment lines
    * @param tagverb 0: no tags, 1: only standard names 2:all HIP numbers
    * @param tagCntr 0=billboard, 1=fixed, 2=facing
    * @param position Distance of the default view point from the center
    * @since 2024-03-01
    */
    public void writeX3d(String ofname, String sname, String cmt, final int tagverb,
                final int tagCntr, final double position)
    {
        try
        {
            StringWriter v = new StringWriter() ;
            BufferedWriter b = new BufferedWriter(v) ;
            final String eol = System.getProperty("line.separator") ;

            b.write( x3dHdr(eol,sname) );

            b.write(cmt);
            b.newLine() ;

            writeScene(b, tagverb, tagCntr, position) ;
            b.write("</X3D>"); /* end of file, closes what x3dHdr started */
            b.newLine() ;

            b.flush() ;
            {
                try
                {
                    Txt2File.writeTxt(v.toString(),ofname) ;
                }
                catch ( Exception ex)
                {
                    System.out.println(ofname + " not written") ; /*does not matter*/
                }
            }
        }
        catch( Exception ex)
        {
            ex.printStackTrace() ;
        }
    } /* writeX3d */

    /** Create one Html file which shows the spheres.
```

```
* @param ofname Name of the output file
* @param sname Name of the associated assembly/part
* @param cmt Additional comment lines
* @param tagverb 0: no tags, 1: only standard names 2:all HIP numbers
* @param tagCntr 0=billboard, 1=fixed, 2=facing
* @param position Distance of the default view point from the center
* @since 2024-03-24
*/
public void writeHtml(String ofname, String sname, String cmt, final int tagverb,
            final int tagCntr, final double position)
{
    try
    {
        StringWriter v = new StringWriter() ;
        BufferedWriter b = new BufferedWriter(v) ;
        final String eol = System.getProperty("line.separator") ;

        b.write( htmlHdr(eol,sname) );

        b.write("<body style='background-color:black;'>");
        b.newLine() ;
        b.write("<X3D>");
        b.write(cmt);
        b.newLine() ;

        writeScene(b, tagverb, tagCntr, position) ;
        b.write("</X3D>"); /* end of file, closes what htmlHdr started */
        b.newLine() ;
        b.write("</body>");
        b.newLine() ;
        b.write("</html>");

        b.flush() ;
        {
            try
            {
                Txt2File.writeTxt(v.toString(),ofname) ;
            }
            catch ( Exception ex)
            {
                System.out.println(ofname + " not written") ; /*does not matter*/
            }
        }
    }
    catch( Exception ex)
    {
        ex.printStackTrace() ;
    }
} /* writeHtml */

/** Create HTML header
* @param sname The title entry.
* @param eol The end-of-line terminator; a line feed in VRML97 and
*  not necessarily the end-of-line terminator of the operating system.
* @return The header with the title and info lines.
* @since 2024-03-24
*/
String htmlHdr(final String eol, final String sname)
{
    String hdr ;
    hdr = new String("<html>" + eol) ;
    hdr += "<head>"+ eol;
    hdr += "<meta name=\"creator\" content=\"" + getClass().getName() +"\"/>" + eol;
    hdr += "<meta name=\"created\" content=\"" + (new Date()).toString() + "\"/>" + eol;
    hdr += "<meta name=\"reference\" content=\"https://vixra.org/abs/1508.0070\"/>" + eol;
    hdr += "<title>" + sname+"</title>" + eol;
    hdr += "<script type='text/javascript' src='https://www.x3dom.org/download/x3dom.js'> </script>" + eol;
    hdr += "<link rel='stylesheet' type='text/css' href='https://www.x3dom.org/download/x3dom.css'/>" + eol;
    hdr += "</head>"+ eol;
    return hdr ;
} /* x3dHdr */

/** Create x3d XML header
* @param sname The title entry.
* @param eol The end-of-line terminator; a line feed in VRML97 and
*  not necessarily the end-of-line terminator of the operating system.
* @return The header with the title and info lines.
* @since 2024-03-01
*/
String x3dHdr(final String eol, final String sname)
{
    String hdr ;
    hdr = new String("<?xml version=\"1.0\" encoding=\"utf-8\"?>" + eol) ;
    hdr += "<X3D xmlns:xsd=\"http://www.w3.org/2001/XMLSchema-instance\""
            + " xsd:noNamespaceSchemaLocation=\"http://www.web3d.org/specifications/x3d-3.1.xsd\">"+ eol;
    hdr += "<head>"+ eol;
    hdr += "<meta name=\"title\" content=\"" + sname+"\"/>" + eol;
    hdr += "<meta name=\"creator\" content=\"" + getClass().getName() +"\"/>" + eol;
    hdr += "<meta name=\"created\" content=\"" + (new Date()).toString() + "\"/>" + eol;
    hdr += "<meta name=\"reference\" content=\"https://vixra.org/abs/1508.0070\"/>" + eol;
    hdr += "</head>"+ eol;
    return hdr ;
} /* x3dHdr */

/** Create the <Scene/> major part of the X3D file
* @param b Name of the byte stream to get the new lines.
* @param tagverb 0: no tags, 1: only standard names 2:all HIP numbers
* @param tagCntr 0=billboard, 1=fixed, 2=facing
* @param position Distance of the default view point from the center
* @since 2024-03-01
*/
public void writeScene(BufferedWriter b, final int tagverb, final int tagCntr, final double position)
{
    try
    {
        b.write("<Scene>");
        b.newLine() ;

        /* apparently disruptive if used in a https://www.x3dom.org/download/x3dom.js
        * b.write("<Viewpoint position=\"0 0 " + position + "\" orientation=\"0 0 1 0.0\"/>");
        * b.newLine() ;
        */

        if ( tagCntr ==0 && tagverb > 0 )
        {
            for(int c=0 ; c < sphs.size() ; c++)
            {
                Sphere thisp = sphs.elementAt(c) ;
                /* print the tag if either tagverb >=2 or if it does not start with one
                * of the boring HD or HIP numbers.
                */
                boolean doPrint = ( tagverb > 1 )
                    || ( ! thisp.tag.startsWith("HIP") && ! thisp.tag.startsWith("HD") ) ;

                if (thisp.tag.length() == 0 )
                    /* optimization of output: empty tag, nothing to print on the billboard */
                    doPrint = false ;
                if ( ! doPrint)
                    continue ;

                b.write("<Transform ") ;

                /* the VRML coordinate system is x=right, y=up and z=towards viewer, so
                * we redefine the original coordinate set (x,y,z) -> (x,z,-y).
                */
                b.write("translation=\"" + thisp.coo[0] + " " + thisp.coo[2] + " "
                    + (-thisp.coo[1]) +"\">") ;
                b.newLine() ;

                b.write("\t<Billboard axisOfRotation=\"0 0 0\">") ;
                b.newLine() ;


                b.write("\t<Shape>") ;
                b.write("\t<Text string=\"" + thisp.tag +"\">") ;
                b.write(" <FontStyle size=\"" + thisp.radius*1.5 + "\" justify=\"END\" solid=\"true\"/>") ;
                b.write(" </Text>") ;
                b.write("\t</Shape>") ;
                b.newLine() ;
                b.write("\t</Billboard>") ;
                b.newLine() ;

                b.write("</Transform>") ;  /* end of Transform */
                b.newLine() ;
            }
        }

        b.write("<Transform>") ;
        b.newLine() ;

        if( sphs != null)
        {
            for(int c=0 ; c < sphs.size() ; c++)
            {
                Sphere thisp = sphs.elementAt(c) ;
                b.write("<Transform ") ;
                /* the VRML coordinate system is x=right, y=up and z=towards viewer, so
                * we redefine the original coordinate set (x,y,z) -> (x,z,-y).
                */
                b.write("translation=\"" + thisp.coo[0] + " " + thisp.coo[2] + " " + (-thisp.coo[1])+"\"" ) ;
                if ( tagCntr == 2)
                {
                    final double[] rang = thisp.perp() ;
                    /* again the calculation within Point3d.pere() uses standard cartesian
                    * coordinates which must be redefined in the VRML97 coordinate system.
                    */
                    b.write("rotation=" + rang[0] + " " + rang[2] + " "
                        + (-rang[1]) + " " + rang[3]+"\"" ) ;
                }
                b.write(">" ) ;
                b.newLine() ;

                b.write("<Shape>") ;
                b.newLine() ;

                /* apprently some viewers require to put the appearance
                * before the object!
                */
                b.write("\t<Appearance>") ;
                b.newLine() ;
                b.write("\t\t<Material") ;
```

```
                b.write(" emissiveColor=\"" + thisp.rgbEmit[0] + " "
                    + thisp.rgbEmit[1] + " " + thisp.rgbEmit[2] +"\"" ) ;
                b.write(" diffuseColor=\"" + thisp.rgb[0] + " " + thisp.rgb[1] + " " + thisp.rgb[2] +"\"") ;
                b.write("/>") ;
                b.newLine() ;
                b.write("\t</Appearance>") ;

                b.newLine() ;

                /* shape the center as a box with flat shape along the ecliptic
                */
                if ( thisp.len() < 1.e-12 )
                    b.write("\t<Box size=\"" + (2.*thisp.radius) + " " + thisp.radius/2 + " " + thisp.radius + "\"/
                else
                    b.write("\t<Sphere radius=\"" + thisp.radius + "\"/>") ;
                b.newLine() ;

                if ( tagverb >0 && tagCntr > 0)
                {
                    if ( tagverb > 1)
                        b.write("\t<Text string=\"" + thisp.tag +"\"></Text>" ) ;
                    else if ( ! thisp.tag.startsWith("HIP") && ! thisp.tag.startsWith("HD") )
                        b.write("\t<Text string=\"" + thisp.tag +"\"></Text>" ) ;
                }
                else
                    b.write("\t <!-- " + thisp.tag + "-->") ;

                b.newLine() ;
                b.write("</Shape>") ; // end children
                b.newLine() ;

                b.write("</Transform>") ; //end transform
                b.newLine() ;
            }
        }

        b.newLine() ;

        b.write("</Transform>") ; /* end of Transform */
        b.newLine() ;

        b.write("</Scene>"); /* end of Scene */
        b.newLine() ;
    }
    catch( Exception ex)
    {
        ex.printStackTrace() ;
    }
} /* writeScene */

/** Create one VRML97 file which shows the spheres.
* @param ofname Name of the output file
* @param sname Name of the associated assembly/part
* @param cmt Additional comment lines
* @param tagverb 0: no tags, 1: only standard names 2:all HIP numbers
* @param tagCntr 0=billboard, 1=fixed, 2=facing
* @param position Distance of the default view point from the center
* @since 2011-08-25
* @since 2023-10-17 don't emit the length tag for texts
*/
public void writeVrml(String ofname, String sname, String cmt, final int tagverb,
            final int tagCntr, final double position)
{
    try
```

```
{
    StringWriter v = new StringWriter() ;
    BufferedWriter b = new BufferedWriter(v) ;
    final String eol = System.getProperty("line.separator") ;

    b.write( vrmlHdr(eol,sname) );

    b.write(cmt);
    b.newLine() ;

    /* Viewpoint { position x y z orientation ax ay az angl fieldOfView 0.sth description "Camra 1" }
    */
    b.write("Viewpoint { position 0 0 " + position + " orientation 0 0 1 0.0 }");
    b.newLine() ;

    if ( tagCntr ==0 && tagverb > 0 )
    {
        for(int c=0 ; c < sphs.size() ; c++)
        {
            Sphere thisp = sphs.elementAt(c) ;
            /* print the tag if either tagverb >=2 or if it does not start with one
            * of the boring HD or HIP numbers.
            */
            boolean doPrint = ( tagverb > 1 )
                || ( ! thisp.tag.startsWith("HIP") && ! thisp.tag.startsWith("HD") ) ;

            if (thisp.tag.length() == 0 )
                /* optimization of output: empty tag, nothing to print on the billboard */
                doPrint = false ;

            if ( ! doPrint)
                continue ;

            b.write("Transform { ") ;
            b.newLine() ;

            /* the VRML coordinate system is x=right, y=up and z=towards viewer, so
            * we redefine the original coordinate set (x,y,z) -> (x,z,-y).
            */
            b.write("translation " + thisp.coo[0] + " " + thisp.coo[2] + " " + (-thisp.coo[1]) ) ;
            b.newLine() ;

            b.write(" children [ Billboard { axisOfRotation 0 0 0 children [") ;
            b.newLine() ;

            b.write("\tShape { geometry Text { string [ \"" + thisp.tag +"\" ] } } " ) ;
            b.write("] } ]") ;
            b.newLine() ;

            b.write("}") ;  /* end of Transform */
            b.newLine() ;
        }
    }

    b.write("Transform {") ;
    b.newLine() ;
    b.write("children") ;
    b.newLine() ;
    b.write("[") ;
    b.newLine() ;

    /* http://gun.teipir.gr/VRML-amgem/spec/part1/nodesRef.html#Shape */
    if( sphs != null)
    {
        for(int c=0 ; c < sphs.size() ; c++)
        {
            Sphere thisp = sphs.elementAt(c) ;
            b.write("Transform { ") ;
            b.newLine() ;
            /* the VRML coordinate system is x=right, y=up and z=towards viewer, so
            * we redefine the original coordinate set (x,y,z) -> (x,z,-y).
            */
            b.write("translation " + thisp.coo[0] + " " + thisp.coo[2] + " " + (-thisp.coo[1]) ) ;
            b.newLine() ;
            if ( tagCntr == 2)
            {
                final double[] rang = thisp.perp() ;
                /* again the calculation within Point3d.pere() uses standard cartesian
                * coordinates which must be redefined in the VRML97 coordinate system.
                */
                b.write("rotation " + rang[0] + " " + rang[2] + " "
                    + (-rang[1]) + " " + rang[3] ) ;
                b.newLine() ;
            }

            b.write("children [ Shape {") ;
            b.newLine() ;

            /* shape the center as a box with flat shape along the ecliptic
            */
            if ( thisp.len() < 1.e-12 )
                b.write("\tgeometry Box { size " + (2.*thisp.radius) + " "
                            + thisp.radius/2 + " " + thisp.radius + " } ") ;
            else
                b.write("\tgeometry Sphere { radius " + thisp.radius + " } ") ;
            b.newLine() ;
            b.write("\tappearance Appearance { material Material { ") ;
            b.write(" emissiveColor " + thisp.rgbEmit[0] + " " + thisp.rgbEmit[1]
                        + " " + thisp.rgbEmit[2] ) ;
            b.write(" diffuseColor " + thisp.rgb[0] + " " + thisp.rgb[1] + " " + thisp.rgb[2] ) ;
            b.write(" } }") ;
            b.write("\t}") ;

            b.newLine() ;

            if ( tagverb >0 && tagCntr > 0)
            {
                if ( tagverb > 1)
                    b.write("\tShape { geometry Text { string [ \"" + thisp.tag +"\" ]  } } " ) ;
                else if ( ! thisp.tag.startsWith("HIP") && ! thisp.tag.startsWith("HD") )
                    b.write("\tShape { geometry Text { string [ \"" + thisp.tag +"\" ] } } " ) ;
            }
            else
                b.write("\t# " + thisp.tag) ;

            b.newLine() ;
            b.write("\t] ") ; // end children
            b.newLine() ;

            b.write("}") ; //end transform
            b.newLine() ;
        }
    }

    b.write("]") ;  /* end of children */
    b.newLine() ;

    b.write("}") ; /* end of Transform */
```

```
            b.newLine() ;

            b.flush() ;
            {
                try
                {
                    Txt2File.writeTxt(v.toString(),ofname) ;
                }
                catch ( Exception ex)
                {
                    System.out.println(ofname + " not written") ; /*does not matter*/
                }
            }
        }
        catch( Exception ex)
        {
            ex.printStackTrace() ;
        }
    } /* writeVrml */

    /** Create VRML header
    * @param sname The title entry.
    * @param eol The end-of-line terminator; a line feed in VRML97 and
    *  not necessarily the end-of-line terminator of the operating system.
    * @return The header with the title and info lines.
    * @since 2012-02-18
    */
    String vrmlHdr(final String eol, final String sname)
    {
        String hdr ;
        hdr = new String("#VRML V2.0 utf8" + eol) ;
        hdr += "WorldInfo {"+ eol;
        hdr += "title \""+  sname+"\"" + eol;
        hdr += "info [" ;
        hdr += "\"" + getClass().getName() + "\"," ;
        hdr += "\"" + (new Date()).toString() + "\", \"" + System.getenv("USERNAME") +"\"";
        hdr += "]" + eol;
        hdr += "}"+ eol;
        return hdr ;
    } /* vrmlHdr */

} /* SphereSet */
```

11. File de/mpg/mpia/hip/Txt2File.java

```
package de.mpg.mpia.hip ;

import java.io.* ;
import java.awt.* ;
import java.awt.event.* ;
import javax.swing.* ;

/** An interface which puts a String into a text file or a byte sequence into a binary file.
* The strategy is to provide two ways of writing: one supported by static methods
* which force the contents into the file, not asking for permission to overwrite,
* the other one with a ctor that displays a confirmation dialog if these files exist.
* @author Richard J. Mathar
* @since 2011-06-09
*/
public class Txt2File {
    /** Name of the file that will receive the output.
    */
    private String fname ;

    /** True if the files are opened in append mode by default. False if overwrite mode.
    * In most applications, in particular .stl and .plt files, appending is not wise, so
    * the default is declared to be false, and the applications ought initiate the
    * instance of the class with last parameter equal to true to prevent overwriting.
    */
    private static boolean PRINT_APPEND = false;

    /** Ctor providing a text and an output file name.
    * This is the interface to the safe (interactive) mode that asks before overwriting.
    * @param logTxt What is to be placed into the file
    * @param appName Name of the application for use in window headers
    * @param foutname Fully qualified file name targeted for output.
    */
    public Txt2File(final String logTxt, final String appName, final String foutname)
    {
        try
        {
            writeTxt(logTxt,foutname) ;
        }
        catch(Exception ex)
        {
            System.out.println(ex) ;
        }
    }

    /** Place a string into an ASCII file.
    * Batch mode without any user interaction.
    * @param logtxt The text to be appended. This text includes any cr and lf's.
    * @param fName The file to be changed. Overwritten or appended if existing (depending on the
    *   the configuration variable PRINT_APPEND), created if non-existing.
    * @throws IOException If the writing did not succeed.
    * @since 2013-04-15 Create parent directory if not yet existing.
    */
    public static void writeTxt(final String logtxt, final String fName) throws IOException
    {
        if ( fName.length() > 0 && logtxt.length() > 0)
        {
            /* create parent directory if not yet existing
            */
            File parDir = new File(fName).getParentFile() ;
            /* parDir may be null if this name is just the single name and does not
            * mention its parent.
            */
            if ( parDir != null)
            {
                if ( ! parDir.exists() )
                    parDir.mkdirs() ;
            }

            FileWriter f = new FileWriter(fName,PRINT_APPEND) ;
            BufferedWriter r = new BufferedWriter(f) ;
            r.write(logtxt) ;
            r.close() ;
            f.close() ;
        }
        } /* writeTxt */

} /* Txt2File */
```

12. File hipparcos/tools/ident5.txt

Volume 13 of the Hipparcos catalogue contains a list of common names for variable stars which are downloaded from `ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident5.doc.gz`. The file is decompressed, the vertical bars are removed and the entries are sorted along increasing Hipparcos ID numbers with

```
gunzip idenet5.doc.gz
dos2unix < ident5.doc | tr "|" " " | awk '{print $2,$1}' | sort -n > ident5.txt
```

The following listing shows only the first 30 lines of a total of 6390 lines of the file:

```
8 Z_Peg
40 V463_Cep
63 CG_And
76 V401_And
99 WZ_Cas
109 DR_Psc
154 YY_Psc
181 V822_Cas
215 V396_Cep
226 RU_Scl
262 TW_And
270 V397_Cep
274 V639_Cas
302 V398_Cep
316 NN_Peg
320 UU_Cet
328 AT_Scl
336 V739_Cas
344 SV_And
363 SU_And
386 V399_Cep
390 IX_Cas
418 V567_Cas
443 BC_Psc
457 BH_Phe
500 BI_Phe
516 SW_Scl
518 V640_Cas
520 CE_Cet
523 CQ_Tuc
```

13. File hipparcos/tools/ident2.txt

The association of Hipparcos ID's with Henry-Draper numbers [7–9] is established by getting the number pairs from `ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident2.doc.gz` and sorting them along the Hipparcos numbers:

```
gunzip ident2.doc.gz
dos2unix < ident2.doc | tr "|" " " | awk '{print $2,$1}' | sort -n > ident2.txt
```

The following listing shows only the first 30 out of 99133 lines of the file:

```
1 224700
2 224690
3 224699
4 224707
5 224705
8 224709
9 224708
10 224717
11 224720
12 224715
```

```
13 224728
14 224726
15 236267
17 224732
19 224721
20 224723
21 224724
22 224735
23 224742
24 224746
25 224750
26 224744
27 224748
28 224749
29 224751
30 224757
31 224760
32 224756
33 224743
34 224758
```

14. File hipparcos/tools/ident4.txt

Bayer-Hamsteed names are obtained by downloading the associated table of volume 13 of the catalogue from `ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident4.doc.gz`, decompressing the file and sorting the content along the Hipparcos numbers. Some Hipparcos numbers appear multiple times: 677, 1067, 1168, 1366 and so on; these duplicates are removed, so the `ident4.doc` file contains 4440 lines but `ident4.txt` only 3264.

```
gunzip ident4.doc.gz
dos2unix < ident4.doc | tr "|" " " | awk '{print $2,$1}' | sort -u -k 1n > ident4.txt
```

The following listing shows only the first 30 lines of the file:

```
88 tau_Phe
122 the_Oct
145 29_Psc
154 30_Psc
171 85_Peg
183 zet_Scl
186 31_Psc
194 c_Psc
301 2_Cet
330 9_Cas
355 3_Cet
443 33_Psc
476 86_Peg
531 10_Cas
664 5_Cet
677 alf_And
729 87_Peg
746 bet_Cas
761 kap01_Scl
765 eps_Phe
813 34_Psc
814 gam03_Oct
841 22_And
910 6_Cet
930 kap02_Scl
950 the_Scl
1067 gam_Peg
1086 23_And
```

```
1168 chi_Peg
1170 7_Cet
```

15. File hipparcos/tools/StarRGB.txt

The file `StarRGB.txt` translates spectral types into RGB triples in the range 0 to 255. In practise we take Noll's proposal of star colors from `http://www.isthe.com/chongo/tech/astro/HR-temp-mass-table-byhrclass.html`, eliminate lines that contain `Ib` or `Ic` or `Ia0`, and substitute all `Ia0` by `I`. The lines are sorted such that the longest strings appear first, because the program will scan the file and return as soon as the Hipparcos color classification starts with a string in the first portion of the `StarRGB.txt` line.

```
lynx -dump http://www.isthe.com/chongo/tech/astro/HR-temp-mass-table-byhrclass.html
   | awk '{if (NF==12 && $12 < 256) printf("%s\t%s %s %s\n",$1,$10,$11,$12)}' > tmp.txt
egrep -v '(Ib|Ic|Ia0)' < tmp.txt | sed 's/Ia/I/' | sort -r > StarRGB.txt
```

The following listing shows only the first 30 lines of the file:

```
WN9VI        154 178 255
WN9V        154 178 255
WN9IV        155 177 255
WN9III        158 177 255
WN9II        160 186 255
WN9I          164 185 255
WN8VI        157 177 255
WN8V        157 177 255
WN8IV        153 176 255
WN8III        157 178 255
WN8II        158 183 255
WN8I          162 183 255
WN7VI        157 177 255
WN7V        157 177 255
WN7IV        152 175 255
WN7III        158 177 255
WN7II        156 181 255
WN7I          160 181 255
WN6VI        162 184 255
WN6V        162 184 255
WN6IV        151 174 255
WN6III        156 175 255
WN6II        155 179 255
WN6I          157 178 255
WN5VI        155 176 255
WN5V        155 176 255
WN5IV        150 172 255
WN5III        154 174 255
WN5II        153 177 255
WN5I          155 176 255
```

[1] M. A. C. Perryman, L. Lindegren, J. Kovalevsky, E. Høg, U. Bastian, *et al.*, The Hipparcos catalogue, Astron. Astrophys. **323**, L49 (1997).
[2] F. P. A. Vogt, C. I. Owen, L. Verdes-Montenegro, and S. Borthakur, Advanced data visualization in astrophysics: the X3D pathway, Astroph. J. **818**, 115 (2016).
[3] R. Carey and G. Bell, *The Annotated VRML 97 Reference* (Addison Wesley, 1999).
[4] ISO/IEC, *VRML97 Functional specification and VRML97 External Authoring Interface (EAI)* (2006).
[5] C. J. Fluke, D. G. Barnes, and N. T. Jones, Interchanging interactive 3D graphics for astronomy, Publ. Astron. Soc. Austr. **26**, 64 (2009).
[6] T. J. Lukka and J. A. Stewart, *FreeWrl* (2015).
[7] A. J. Cannon and E. C. Pickering, The henry draper catalogue 0h, 1h , 2h and 3h, Ann. Harvard Coll. Obs. **91**, 1 (1918).
[8] A. J. Cannon and E. C. Pickering, The henry draper catalogue 19h and 20h, Ann. Harvard Coll. Obs. **98**, 1 (1923).
[9] A. J. Cannon and E. C. Pickering, The henry draper catalogue 21h, 22h, and 23h, Ann. Harvard Coll. Obs. **99**, 1 (1924).