`hip2wrl`: A Java Program to Represent a Hipparcos Star Collection as a VRML97 File

Richard J. Mathar*
Hoeschstr. 7, 52372 Kreuzau, Germany
(Dated: August 10, 2015)

A Java program is presented which extracts star positions from the Hipparcos main catalogue and places them into a sphere collection rendered in a VRML97 file. The main options to the executable are a cut-off distance to some center of the scene (the sun by default) and a density of 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 format of the Virtual Reality Modeling Language (VRML) [2, 3]. 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.

The advantages of that particular 3D format are [5]

- independence from operating systems and computer platforms,
- roality-free use without license fees,
- access to the documented standard without the common ISO pay wall, and
- in particular an embedded text positioning system that helps to keep track of which star is where while changing the viewing perspectives.

Alternative representations with equivalent advantages are

- the `gnuplot` system which allows rotation of circles presented by its `splot` command, and
- the MDL Molfiles.

B. Parameter Selection

The 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 VRML 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.

* http://www.mpia.de/~mathar

FIG. 1. Screen shot looking at a VRML97 file generated with a cut-off radius of 30 light years around the sun, as presented by `freewrl` [4].

- 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 14), the sphere is shown in grey.

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` *[*`-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 7 options may be omitted. The options define the following parameters:

- `-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 VRML file. The run time of the program and the VRML 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 VRML file that will be created. If the option is not used, the name `hip_main.wrl` 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 2). If the Hipparcos star does not fall into that category, the Bayer/Flamsteed name is looked up (Appendix A 13). 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 some 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 12), 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 VRML file is labeled.

  Note that some VRML viewers 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 VRML scene. The default is `0`.

  - The value of `0` uses the 'billboard' option of the VRML 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 VRML97 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 hipparcos/tools/DelimitedLine.java

```
1  package hipparcos.tools;
2
3  import java.io.*;
4  import java.lang.*;
5  import java.util.*;
6
7  /** Wraps up a line in s Stringtokenizeer using a given delimiter
8  provide usefull casting methods to get back doubles etc.
9  */
10 public class DelimitedLine {
11     protected StringTokenizer line;
12
13     /** Ctor
14     * @param line the string to be parsed as alime
15     * @param delim the delimiter between the fields in th eline
16     */
17     public DelimitedLine(String line, char delim) {
18         this.line = new StringTokenizer(line,new String(delim+""));
19     }
20
21     /** Get the next token in the sequence of fields.
22     * @return The string in the next entry that was extracted.
23     * Leading or trailing white space is removed.
24     * @throws Exception If the format in the data input stream did not match the expectation.
25     */
26     protected String getNext() throws Exception {
27         return (line.nextToken().trim());
28     }
29
30     /** Get the next token in the sequence of fields
31     * @return Get the next field between the delimiters.
32     * @throws Exception If the format in the data input stream did not match the expectation.
33     */
34     public String getNextString() throws Exception {
35         return (getNext());
36     }
37
38     /** Interpret the next token as an integer
39     * @return The integer that was extracted.
40     * @throws Exception If the format in the data input stream did not match the expectation.
41     */
42     public int getNextInt() throws Exception {
43         return (new Integer(getNext()).intValue());
44     }
45
```

```
46      /** Interpret the next token as a double precision number.
47      * @return The doubles that was extracted.
48      * @throws Exception If the format in the data input stream did not match the expectation.
49      */
50      public double  getNextDouble() throws Exception {
51          String tmp = getNext();
52          if (tmp.length() == 0) {
53              throw (new Exception ());
54          }
55          return (new Double(tmp).doubleValue());
56      }
57
58      /** parse the string as a set of doubles and return it in an array
59      * @return The sequence of doubles that were parsed.
60      * @throws Exception If the format in the data input stream did not match the expectation.
61      */
62      public double[] getDoubleArray()  throws Exception {
63          if (line.countTokens() == 0 )
64              throw new Exception("Empty String for array");
65          double[] dnums = new double[line.countTokens()];
66          for (int t=0; t< dnums.length; t++ )  {
67              dnums[t] = getNextDouble();
68          }
69          return dnums;
70      }
71
72      /** parse the string as a set of ints and return it in an array
73      * @return The sequence of integers that were parsed.
74      * @throws Exception If the format in the data input stream did not match the expectation.
75      */
76      public int[] getIntArray()  throws Exception {
77          if (line.countTokens() == 0 )
78              throw new Exception("Empty String for array");
79          int[] inums = new int[line.countTokens()];
80          for (int t=0; t< inums.length; t++ )  {
81              inums[t] = getNextInt();
82          }
83          return inums;
84      }
85
86      /** Skip/ignore 1 or more fields
87      * @param n The positive number (number of fields to skip)
88      * @throws Exception If the format in the data input stream did not match the expectation.
89      * @since 2015-07-17
90      * @author R. J. Mathar
91      */
92      public void skip(int n)  throws Exception {
93          for(int s=0 ; s < n ; s++)
94          {
95              getNext() ;
96          }
97      }
98  }
```

2. File hipparcos/tools/Star.java

```
1   package hipparcos.tools;
2
3   import java.text.*;
4   import java.awt.*;
5   import java.net.*;
6   import java.io.* ;
```

```
7  import de.mpg.mpia.hip.Point3D ;
8
9  /** A class to represent one entry in the hipparcos catalog.
10 * @author William O'Mullane for the Astrophysics Division of ESTEC  - part of the European Space Agency.
11 */
12 public class Star {
13
14     public String type; // H0 or T0 contains H or T
15     public String id; // H1 ot T1
16
17     /** magnituted H5
18     */
19     public double  mag;
20
21     /** alpha in degrees H8
22     */
23     public double alpha;
24
25     /** delta in degrees H9
26     */
27     public double  delta;
28
29     /** paralax in mas, H11
30     */
31     public double  paralax;
32
33     /** proper motion mu(alpha)*cos(delta) H12
34     */
35     public double muAlpha;
36     /** proper motion mu(delta) H13
37     */
38     public double muDelta;
39     public double b_v; //H37
40     public String sptype; // spectral type
41     public boolean inHIPnTYC=false;
42     public boolean inHIP=false;
43
44     /** Default constructor.
45     * Sets the id (field H1) to null.
46     */
47     public Star () {
48         id=null;
49     }
50
51     /** Contruct the star from the delimetd line as it appears in the catalog
52     * @param str The ASCII line of the main catalogue
53     * @throws Exception if the list of strings, integers and doubles in str does not match the main catalogue format.
54     */
55     public Star (String str) throws Exception {
56         DelimitedLine line= new DelimitedLine(str,'|');
57         String skip;
58         type = line.getNextString();
59         id   = line.getNextString();
60         line.skip(3) ;
61         mag  = line.getNextDouble();    //H5
62         line.skip(2) ;
63         alpha = line.getNextDouble();
64         delta = line.getNextDouble();
65         line.skip(1) ;
66         paralax=line.getNextDouble();   //H11
67         try {
68             muAlpha = line.getNextDouble();
69         } catch (Exception e)
70         {};
```

```
71
72          try {
73              muDelta = line.getNextDouble(); //h13
74          } catch (Exception e)
75          {};
76
77          int i=13;
78          if (type.startsWith("T")) {// get T31
79              /* this not for the main catalogue but the Tycho
80              */
81              try {
82                  for (i=13; i<30; i++) {
83                      try {
84                          skip=line.getNextString();
85                      } catch (Exception e) {
86                          System.out.println ("Skip Failed "+ e);
87                      }
88                  }
89                  i++;
90                  String tmpStr=line.getNextString(); //extractHipNo from T31
91                  int tmp= new Integer(tmpStr).intValue();
92                  inHIPnTYC=(tmp >=1);
93              } catch (Exception e) {
94                  inHIPnTYC=false;
95              }
96          } else { inHIP=true; };
97          while (i <  36) {
98              try {
99                  line.skip(1) ;
100                 i++;
101             } catch (Exception e) {
102                 System.out.println ("Skip Failed "+ e);
103             }
104         }
105         b_v = line.getNextDouble(); // H37
106
107         /* skip H38 to H75 inclusive
108         */
109         line.skip(38) ;
110         sptype = line.getNextString(); // H76
111
112     }
113
114     /**
115     * @return The distance to the reference frame (light years)
116     * @since 2015-07-18
117     * @author R. J. Mathar
118     */
119     public double ly()
120     {
121         /* tan (parallax) = au/distance
122         * distance = au/tan(parallax)
123         * distance/ly = (au/ly)/tan(parallax)
124         * Convert mas to as to degrees do radians
125         */
126         final double pax = paralax/1000.0/3600.0/Constants.degc ;
127         return 1.0/(Math.tan(pax)*Constants.lyau) ;
128     }
129
130     /**
131     * Distance to a referencd point.
132     * @param ref The reference point.
133     * @return The distance in light years.
134     * @since 2015-08-09
```

```
135       * @author R. J. Mathar
136       */
137       public double ly(final Point3D ref)
138       {
139           return posit().distance(ref) ;
140       }
141
142       /**
143       * Construct the Cartesian coordinates where the ICRS ecliptic is the (x,y) plane.
144       * @return The three components of the position in units of light years.
145       * @since 2015-07-18
146       * @author R. J. Mathar
147       */
148       public Point3D posit()
149       {
150           final double r = ly() ;
151           /* convert alpha and delta on the fly from degrees to radians
152           */
153           final double x = r*Math.cos(alpha/Constants.degc)*Math.cos(delta/Constants.degc) ;
154           final double y = r*Math.sin(alpha/Constants.degc)*Math.cos(delta/Constants.degc) ;
155           final double z = r*Math.sin(delta/Constants.degc) ;
156           return new Point3D(x,y,z) ;
157       }
158
159       /**
160       * @param scaleMag if true scale the RGB color with some invers of the magnitude.
161       * @return an integer triple of 0.0..1.0 RGB colors
162       */
163       public float[] rgbCol(final boolean scaleMag)
164       {
165           float[] rgb = new float[3] ;
166           /* default (if not found..
167           */
168           rgb[0]=rgb[1] =rgb[2] = 0.0f ;
169
170           String cleanSptype = sptype;
171
172           /* take what is after a slash ...
173           int idx = cleanSptype.indexOf("/") ;
174           if ( idx >= 0)
175               cleanSptype = cleanSptype.substring(idx+1) ;
176           */
177           /* take what is before a slash ...
178           */
179           int idx = cleanSptype.indexOf("/") ;
180           if ( idx >= 0)
181               cleanSptype = cleanSptype.substring(0,idx) ;
182
183           /* replace 1.5, 2.5, 3.5 etc by straight integers
184           */
185           idx = cleanSptype.indexOf(".5") ;
186           if ( idx >= 0)
187               cleanSptype = cleanSptype.substring(0,idx) + cleanSptype.substring(idx+2) ;
188
189           /* delete anything after a colon
190           */
191           idx = cleanSptype.indexOf(":") ;
192           if ( idx >= 0)
193               cleanSptype = cleanSptype.substring(0,idx) ;
194
195           /* delete anything after a triple dots
196           */
197           idx = cleanSptype.indexOf("...") ;
198           if ( idx >= 0)
```

```
199             cleanSptype = cleanSptype.substring(0,idx) ;
200
201         /* delete anything after a plus
202         */
203         idx = cleanSptype.indexOf("+") ;
204         if ( idx >= 0)
205             cleanSptype = cleanSptype.substring(0,idx) ;
206
207         /* simple types like G0 or A2... converted to G0III etc
208         */
209         String extSptype =cleanSptype;
210         if ( cleanSptype.length() == 2 )
211         {
212             if ( sptype.startsWith("D") )
213                 extSptype = cleanSptype + "3" ;
214             else
215                 extSptype = cleanSptype + "III" ;
216         }
217
218         /* http://www.isthe.com/chongo/tech/astro/HR-temp-mass-table-byhrclass.html
219         */
220         URL rgbtxt = getClass().getResource("StarRGB.txt") ;
221         try
222         {
223             FileReader istream = new FileReader(rgbtxt.getFile()) ;
224             BufferedReader dstream = new BufferedReader(istream) ;
225             for(;;)
226             {
227                 String lin = dstream.readLine() ;
228                 if ( lin == null)
229                     break ;
230                 /* split at tab */
231                 String[] linFields = lin.split("\t") ;
232                 if ( extSptype.toUpperCase().startsWith(linFields[0]) )
233                 {
234                     for( int c=0 ; c < 3 ; c++)
235                     {
236                         /* rgb values are originaly in the range 0..255.
237                         * Convert them to VRML units by division through 256.
238                         */
239                         rgb[c] = (new Short(linFields[1].substring(4*c,4*c+3))).floatValue()/256.0f ;
240                     }
241                     break;
242                 }
243
244             }
245         }
246         catch (Exception ex)
247         {
248             System.err.println(ex) ;
249             ex.printStackTrace() ;
250         }
251
252         if ( rgb[0] == 0.0f && rgb[1] == 0.0f && rgb[2] == 0.0f)
253         {
254             rgb[0] = rgb[1] = rgb[2] = 0.6f ;
255         }
256
257         if ( scaleMag)
258             for( int c=0 ; c < 3 ; c++)
259             {
260                 /* Star magnitudes are in the range >= -1.44, so we take this
261                 * as the "origin" of the luminosity: mag+1.44.
262                 */
```

```
263                 rgb[c] /= 1.54+mag ;
264             }
265
266         return rgb ;
267     } /* rgbCol */
268
269     /**
270     * @return The magnitude H5
271     */
272     public double getMag() {
273         return mag;
274     }
275
276     /**
277     * @return The alpha coordinate (deg) H8
278     */
279     public double getAlpha() {
280         return alpha;
281     }
282
283     /**
284     * @return The delta coordinate (deg) H9
285     */
286     public double getDelta() {
287         return delta;
288     }
289
290     /**
291     * @return The astrometric paralax (mas) H11
292     */
293     public double getParalax() {
294         return paralax;
295     }
296
297
298     /**
299     * @return Hipparcos or Tycho id (as a string)
300     */
301     public String getId() {
302         return id;
303     }
304
305     /**
306     * @return First field in the catalogue. "H" or "T".
307     */
308     public String getType() {
309         return type;
310     }
311
312     /**
313     * @return True if the star is in the Hipparcos catalogue. False if in the Tycho catalogue.
314     */
315     public boolean inHIP() {
316         return inHIP;
317     }
318
319     /**
320     * @param years The number of years extrapolated from the epoch.
321     * @return The total change in alpha due to proper motion (deg)
322     */
323     public double getMuAlpha(int years) {
324         return (years*(muAlpha/3600000));
325     }
326
```

```
327
328     /**
329     * Translate a HIP number to a Bayer/Flamsteed star name of volume 13 of the main catalogue
330     * @return A more common name. This is empty if there is none.
331     * @since 2015-07-27
332     */
333     public String hip2ident4()
334     {
335         final int hipid = (new Integer(id).intValue());
336         return hip2ident4(hipid) ;
337     }
338
339     /**
340     * Translate a HIP number to a Variable star name of volume 13 of the main catalogue
341     * @return A name of table ID5. This is empty if there is none.
342     * @since 2015-07-29
343     */
344     public String hip2ident5()
345     {
346         final int hipid = (new Integer(id).intValue());
347         return hip2ident5(hipid) ;
348     }
349
350     /**
351     * Translate a HIP number to a Henry-Draper (HD) number name of volume 13 of the main catalogue
352     * @return A name of table ID2. This is empty if there is none.
353     * @since 2015-07-29
354     */
355     public int hip2ident2()
356     {
357         final int hipid = (new Integer(id).intValue());
358         return hip2ident2(hipid) ;
359     }
360
361     /**
362     * Translate a HIP number to a star name of volume 13 of the main catalogue
363     * @param hipid The index of the star in the main catalogue.
364     * @return A more common name refering to the zodiac sign.
365     *   This is empty if there is none (as there are 4440 assignments listed, which
366     *   is only a minor portion of roughly 4 percent of the entire catalogue)
367     * @since 2015-07-27
368     */
369     static public String hip2ident4(int hipid)
370     {
371         /* ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident4.doc.gz
372         * dos2unix < ident4.doc | tr "|" " " | awk '{print $2,$1}' | sort -u -k 1n > ident4.txt
373         */
374         URL idtbl = hipparcos.tools.Star.class.getResource("ident4.txt") ;
375         try
376         {
377             FileReader istream = new FileReader(idtbl.getFile()) ;
378             BufferedReader dstream = new BufferedReader(istream) ;
379             for(;;)
380             {
381                 String lin = dstream.readLine() ;
382                 if ( lin == null)
383                     break ;
384                 /* split at first blank */
385                 String[] linFields = lin.split(" ") ;
386                 final int namid = (new Integer(linFields[0]).intValue());
387                 if ( namid == hipid)
388                     return linFields[1] ;
389                 else if ( namid > hipid)
390                     /* because ident4.txt is sorted numerically in the
```

```
391                     * first field, we can skip the rest because thy will not match either
392                     */
393                     break;
394 
395             }
396         }
397         catch (Exception ex)
398         {
399             System.err.println(ex) ;
400             ex.printStackTrace() ;
401         }
402         return new String() ;
403     }
404 
405     /**
406     * Translate a HIP number to a star name of volume 13 of the main catalogue
407     * @param hipid The index of the star in the main catalogue.
408     * @return A more common name refering to the variable star names.
409     *   This is empty if not covered by table ID5.
410     * @since 2015-07-29
411     */
412     static public String hip2ident5(int hipid)
413     {
414         /* ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident5.doc.gz
415         * dos2unix < ident5.doc | tr "|" " " | awk '{print $2,$1}' | sort -n > ident5.txt
416         */
417         URL idtbl = hipparcos.tools.Star.class.getResource("ident5.txt") ;
418         try
419         {
420             FileReader istream = new FileReader(idtbl.getFile()) ;
421             BufferedReader dstream = new BufferedReader(istream) ;
422             for(;;)
423             {
424                 String lin = dstream.readLine() ;
425                 if ( lin == null)
426                     break ;
427                 /* split at first blank */
428                 String[] linFields = lin.split(" ") ;
429                 final int namid = (new Integer(linFields[0]).intValue());
430                 if ( namid == hipid)
431                     return linFields[1] ;
432                 else if ( namid > hipid)
433                     /* because ident5.txt is sorted numerically in the
434                     * first field, we can skip the rest because thy will not match either
435                     */
436                     break;
437 
438             }
439         }
440         catch (Exception ex)
441         {
442             System.err.println(ex) ;
443             ex.printStackTrace() ;
444         }
445         return new String() ;
446     }
447 
448     /**
449     * Translate a HIP number to a HD number of volume 13 of the main catalogue
450     * @param hipid The index of the star in the main catalogue.
451     * @return The Henry Draper number according to the ID2 table.
452     *   This is 0 if not covered by table ID2.
453     * @since 2015-07-29
454     */
```

```
455     static public int hip2ident2(int hipid)
456     {
457         /* ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident2.doc.gz
458         * dos2unix < ident2.doc | tr "|" " " | awk '{print $2,$1}' | sort -n > ident2.txt
459         */
460         URL idtbl = hipparcos.tools.Star.class.getResource("ident2.txt") ;
461         try
462         {
463             FileReader istream = new FileReader(idtbl.getFile()) ;
464             BufferedReader dstream = new BufferedReader(istream) ;
465             for(;;)
466             {
467                 String lin = dstream.readLine() ;
468                 if ( lin == null)
469                     break ;
470                 /* split at first blank */
471                 String[] linFields = lin.split(" ") ;
472                 final int namid = (new Integer(linFields[0]).intValue());
473                 if ( namid == hipid)
474                     return (new Integer(linFields[1])).intValue() ;
475                 else if ( namid > hipid)
476                     /* because ident2.txt is sorted numerically in the
477                     * first field, we can skip the rest because thy will not match either
478                     */
479                     break;
480 
481             }
482         }
483         catch (Exception ex)
484         {
485             System.err.println(ex) ;
486             ex.printStackTrace() ;
487         }
488         return 0 ;
489     }
490 
491     /**
492     * Translate a HIP number to a common star name
493     * @return A more common name.
494     *  If none is found, the HIPxxxx string with just the hip is returned.
495     */
496     public String hip2name()
497     {
498         final int hipid = (new Integer(id).intValue());
499         return hip2name(hipid) ;
500     }
501 
502     /**
503     * Translate a HIP number to a common star name.
504     * Equivalent to an annex in vol 13 of the catalogue.
505     * @param hip The star number in the main catalogue
506     * @return A more common name.
507     *  If none is found, the HIPxxxx string with just the hip is returned.
508     */
509     static public String hip2name(int hip)
510     {
511         /* http://www.rssd.esa.int/index.php?project=HIPPARCOS&page=common
512         * ftp://cdsarc.u-strasbg.fr/pub/cats/more/HIP/cdroms/tables/ident6.doc.gz
513         */
514         final String[] ident6 = {
515         "Alpheratz",
516         "Caph",
517         "Algenib",
518         "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",
```

```
583         "Alphekka",
584         "Unukalhai",
585         "Antares",
586         "Rasalgethi",
587         "Shaula",
588         "Rasalhague",
589         "Etamin",
590         "Barnard's star",
591         "Kaus Australis",
592         "Vega",
593         "Sheliak",
594         "Nunki",
595         "Albireo",
596         "Campbell's star",
597         "Tarazed",
598         "Altair",
599         "Alshain",
600         "Cyg X-1",
601         "Deneb",
602         "Alderamin",
603         "Enif",
604         "Sadalmelik",
605         "Alnair",
606         "Kruger 60",
607         "Babcock's star",
608         "Fomalhaut",
609         "Scheat",
610         "Markab"
611         } ;
612
613         final int[] hips ={
614         677,
615         746,
616         1067,
617         2081,
618         3179,
619         3419,
620         3829,
621         5447,
622         7588,
623         9640,
624         9884,
625         10826,
626         11767,
627         13847,
628         14135,
629         14576,
630         15863,
631         17702,
632         17851,
633         18543,
634         21421,
635         24186,
636         24436,
637         24608,
638         25336,
639         25428,
640         25606,
641         25930,
642         25985,
643         26311,
644         26727,
645         27366,
646         27989,
```

```
647         30089,
648         30438,
649         31681,
650         32349,
651         33579,
652         36208,
653         36850,
654         37279,
655         37826,
656         46390,
657         49669,
658         50583,
659         53910,
660         54061,
661         57632,
662         57939,
663         58001,
664         59774,
665         60718,
666         60936,
667         62956,
668         63125,
669         63608,
670         65378,
671         65474,
672         65477,
673         67301,
674         68702,
675         68702,
676         68756,
677         69673,
678         70890,
679         71683,
680         72105,
681         72607,
682         76267,
683         77070,
684         80763,
685         84345,
686         85927,
687         86032,
688         87833,
689         87937,
690         90185,
691         91262,
692         92420,
693         92855,
694         95947,
695         96295,
696         97278,
697         97649,
698         98036,
699         98298,
700         102098,
701         105199,
702         107315,
703         109074,
704         109268,
705         110893,
706         112247,
707         113368,
708         113881,
709         113963
710         } ;
```

```
711         /* first attempt: get it from ident6 assignments (common star names)
712         */
713         for(int j=0 ; j < hips.length ; j++)
714         {
715             if ( hips[j] == hip)
716                 return ident6[j] ;
717             else if ( hips[j] > hip)
718                 break ;
719         }
720 
721         /* second backup attempt: get it from the ident4 assignments (Bayer/Flamsteed names)
722         */
723         String i6 = hip2ident4(hip) ;
724 
725         if ( i6.length() > 0 )
726             return i6 ;
727 
728         /* third backup attempt: get it from the ident5 assignments (VS names)
729         */
730         String i5 = hip2ident5(hip) ;
731 
732         if ( i5.length() > 0 )
733             return i5 ;
734 
735         /* fourth backup attempt: get it from the ident2 assignments (HD numbers)
736         */
737         int i2 = hip2ident2(hip) ;
738 
739         if ( i2 > 0 )
740             return new String("HD"+i2) ;
741 
742         /* final refuge is the HIP number itself
743         */
744         return new String("HIP"+hip) ;
745     } /* hip2name */
746 
747 } /* Star */
```

3. File hipparcos/tools/Constants.java

```
1  package hipparcos.tools;
2  
3  /** Constants used in the tools
4  */
5  public class Constants {
6      /** Eulers constant 3.14159..
7      */
8      public static final double PI = Math.PI;
9  
10 
11     /** Pi/2
12     */
13     public static final double cPi = 2.0*Math.atan(1);
14 
15     /** Pi/360
16     */
17     public static final double cPr = cPi/180;
18 
19     /** 180/Pi. Conversion factor radians to degrees
20     */
21     public static final double degc=57.29577951308232087679815481 41 ;
22 
```

```
23      /** Ratio of light year over A.U.
24      * ratio 9,460,730,472,580.8 km / 149 597 870, 700 km
25      * @since 2015-07-18
26      */
27      public static final double lyau = 63241.07708426628026865358318 23 ;
28
29      public static final int vlev = 2;
30
31  } /* Constants */
```

4. File hipparcos/tools/StarFactory.java

```
1   package hipparcos.tools;
2
3   import java.io.*;
4   import java.util.*;
5   import java.net.URL;
6
7   /** For given alpha delta and d get all stars from the database
8   in that area. Optional hipOnly may be given.
9   * @author William O'Mullane for the Astrophysics Division of ESTEC  - part of the European Space Agency.
10  */
11  public class StarFactory  {
12      protected boolean hipOnly=false;
13      protected double alpha=0;
14      protected double delta=0;
15      protected double d=0;
16      protected BufferedReader dstream=null;
17      protected boolean finished=false;
18      protected boolean hipDone=false;
19      protected boolean opened=false;
20      protected boolean disk=false;
21      protected String[] catprogs = {"shipmainra","stycmainra"};
22
23
24      /**
25      * @param hipOnly If true, tycho catalogues are not scanned.
26      */
27      protected StarFactory(boolean hipOnly) {
28          this.hipOnly = hipOnly;
29      } ;
30
31      /**
32      * @param alpha The alpha (RA) coordinate (degrees)
33      * @param delta The delta (DEC) coordinate (degrees)
34      * @param d A value for the box parameter of the URL query.
35      */
36      public StarFactory (double alpha, double delta, double d) {
37          this.alpha = alpha;
38          this.delta = delta;
39          this.d = d;
40          finished = ! openStream();
41      }
42
43      /**
44      * @param alpha The alpha (RA) coordinate (degrees)
45      * @param delta The delta (DEC) coordinate (degrees)
46      * @param d A value for the box parameter of the URL query.
47      * @param hipOnly If true, tycho catalogues are not scanned.
48      */
49      public StarFactory ( double alpha, double delta, double d, boolean hipOnly ) {
50          this.hipOnly = hipOnly;
```

```
51          this.alpha = alpha;
52          this.delta = delta;
53          this.d = d;
54          finished = ! openStream();
55      }
56
57      /**
58      * @param fname The file name of the main catalogue in the local disks
59      * @since 2015-07-17
60      * @author R. J. Mathar
61      */
62      public StarFactory ( String fname)
63      {
64          finished = false ;
65          opened = loadFromDisk(fname) ;
66      }
67
68      /**
69      * @return True if the input stream could be opened successfully.
70      */
71      protected boolean openStream() {
72          opened=true;
73          finished=false;
74          return loadFromDisk(catprogs[0]);
75      }
76
77      /**
78      * @param fname The file name with the ASCII catalog
79      * @return true if sucess, false if unable to open.
80      * @author R. J. Mathar
81      * @since 2015-07-17
82      */
83      protected boolean loadFromDisk(String fname) {
84          try {
85              FileReader istream  = new FileReader(fname);
86              dstream = new BufferedReader(istream);
87          } catch (Exception e) {
88              System.err.println("loadFromDisk " +fname + " "+ e);
89              e.printStackTrace();
90              return false;
91          }
92          return true;
93      }
94
95
96      /** Skip to next PRE tag in html stream.
97      * @throws Exception If reading line of the data input stream did not suceed.
98      */
99      protected void skipToData() throws Exception {
100         String str;
101         boolean found=false;
102         while (!found ) { // skip to pre
103             str=dstream.readLine();
104             found = (str == null) || str.startsWith("<pre>") || str.startsWith("<PRE>");
105         }
106         str=dstream.readLine(); // skip header
107     }
108
109     /** Keep getting stars until none left.
110     * @return The Star in the next line of the input catalogue
111     * @throws NoMoreStars if the catalogue has been entirely read to the end.
112     */
113     public Star getNext() throws NoMoreStars {
114         if (!opened)
```

```
115             openStream();
116
117         if (finished)
118             throw new NoMoreStars();
119
120         boolean found=false;
121         String str;
122         Star star=null;
123         while (!found ) {
124             str=null;
125             try {
126                 str=dstream.readLine();
127             } catch (Exception e) {
128             };
129
130             if (str==null) {
131                 if (disk && catprogs[1]!=null && !hipDone && !hipOnly) {
132                     loadFromDisk(catprogs[1]);
133                     hipDone=true;
134                     try {
135                         str=dstream.readLine();
136                     } catch (Exception tyce) {
137                         tyce.printStackTrace();
138                         throw new NoMoreStars();
139                     }
140                 } else {
141                     finished = true;
142                     try {
143                         dstream.close();
144                     } catch (Exception e) {};
145                     throw new NoMoreStars();
146                 }
147             }
148             else
149             {
150                 if (str.startsWith("</pre>") || str.startsWith("</PRE>")) {
151                     if (hipDone || hipOnly ) {
152                         finished=true;
153                         try {
154                             dstream.close();
155                         } catch (Exception e) {};
156                         throw new NoMoreStars();
157                     } else {
158                         try {
159                             skipToData();
160                             str=dstream.readLine();
161                         } catch (Exception e) {
162                             System.err.println("StarFactory:header "+e);
163                             throw new NoMoreStars();
164                         }
165                         hipDone=true;
166                     }
167                 }
168             }
169             try {
170                 star= new Star(str);
171                 found=true;
172             } catch (Exception e) {
173             }
174         }
175         return star;
176     }
177 } /* StarFactory */
```

5. File hipparcos/tools/NoMoreStars.java

```
1
2  package hipparcos.tools;
3
4  /** Exception tjrown when a factory has no more stars
5  */
6  public class NoMoreStars extends Exception {
7      public NoMoreStars() {};
8      public NoMoreStars(String msg) {
9          super(msg);
10     };
11 }
```

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

```
1  package de.mpg.mpia.hip ;
2
3  import hipparcos.tools.* ;
4
5  /**
6  * A translator of Hipparcos stars into a VRML97 scene.
7  * @since 2015-07-18
8  * @author R. J. Mathar
9  */
10 class hip2wrl {
11
12     /** Define a new center of the scene at some specific star
13     * @param cHip the number in the main catalogue for the new center
14     * @return The location in the global coordinate system
15     */
16     static Point3D locateHip(int cHip, final String catfname)
17     {
18         if ( cHip >0 && cHip <= 118322)
19         {
20             StarFactory cat = new StarFactory(catfname) ;
21             for(;;)
22             {
23                 try
24                 {
25                     Star s = cat.getNext() ;
26                     int thisid = (new Integer(s.getId())).intValue() ;
27                     if ( s.ly() > 0. && thisid == cHip)
28                     {
29                         return  s.posit() ;
30                     }
31                 }
32                 catch (Exception ex)
33                 {
34                     break;
35                 }
36             }
37         }
38         return null ;
39     }
40
41     /**
42     * @param argv Vector of the command line arguments
43     */
44     static public void main(String argv[])
45     {
46         String catfname = new String("./hip_main.dat") ;
```

```
47          String ofname = new String("./hip_main.wrl") ;
48          /* default limit of distance to center is 20 light years
49          */
50          double lyLim = 28.0 ;
51
52          /* standard size of sphere radius is  0.3 (light years)
53          */
54          double rad = 0.3 ;
55
56          int tagVerb = 1 ;
57          /* 0 = billboard, 1=fixed, 2=facing center
58          */
59          int tagCntr = 0 ;
60
61          /* mid point in the scene. By default at the position of the sun.
62          */
63          Point3D cntr = new Point3D() ;
64
65          int cHip = 0 ;
66
67          /* collect options
68          *  -l <limitlightyears>
69          *  -f <filenamecatalog>
70          *  -t [0,1,2]
71          *  -o <outfile>
72          *  -F [0,1,2]
73          *  -r <radius/ly>
74          *  -C <HIPid>
75          */
76          for(int i=0 ; i < argv.length; i++)
77          {
78              if ( argv[i].compareTo("-l") == 0 )
79                  lyLim = (new Double(argv[i+1])).doubleValue() ;
80              else if ( argv[i].compareTo("-r") == 0 )
81                  rad = (new Double(argv[i+1])).doubleValue() ;
82              else if ( argv[i].compareTo("-f") == 0 )
83                  catfname = argv[i+1] ;
84              else if ( argv[i].compareTo("-o") == 0 )
85                  ofname = argv[i+1] ;
86              else if ( argv[i].compareTo("-t") == 0 )
87                  tagVerb = (new Integer(argv[i+1])).intValue() ;
88              else if ( argv[i].compareTo("-F") == 0 )
89                  tagCntr = (new Integer(argv[i+1])).intValue() ;
90              else if ( argv[i].compareTo("-C") == 0 )
91              {
92                  cHip = (new Integer(argv[i+1])).intValue() ;
93              }
94          }
95
96          /* if the user wished to relocate the center
97          */
98          if ( cHip > 0)
99          {
100             Point3D c = locateHip(cHip,catfname) ;
101             if ( c != null)
102                 cntr = c;
103         }
104
105         SphereSet allSphs = new SphereSet() ;
106
107         /* start with the sun
108         */
109         Star sun = new Star() ;
110         if ( sun.ly(cntr) < lyLim)
```

```
111         {
112             sun.id = new String("0") ;
113             sun.paralax = 0.0 ;
114             sun.type = new String("H") ;
115             sun.alpha = sun.delta=0.0 ;
116             sun.sptype = new String("G2V") ;
117 
118             Sphere sph0 = new Sphere(0, 0, 0,rad) ;
119             sph0.rgb = sun.rgbCol(false) ;
120             sph0.rgbEmit = sun.rgbCol(true) ;
121             sph0.tag = new String("Sun") ;
122             allSphs.add(sph0) ;
123         }
124 
125         StarFactory cat = new StarFactory(catfname) ;
126         for(;;)
127         {
128             try
129             {
130                 Star s = cat.getNext() ;
131                 if ( s.ly(cntr) < lyLim & s.ly() > 0.)
132                 {
133                     /* java -cp . de.mpg.mpia.hip.hip2wrl -l 20000 | sort -u
134                     */
135 
136                     Point3D pos = cntr.to(s.posit()) ;
137                     Sphere sph = new Sphere(pos.coo[0], pos.coo[1], pos.coo[2],rad) ;
138                     sph.rgb = s.rgbCol(false) ;
139                     sph.rgbEmit = s.rgbCol(true) ;
140                     sph.tag = s.hip2name() ;
141 
142                     allSphs.add(sph) ;
143                 }
144             }
145             catch (Exception ex)
146             {
147                 break;
148             }
149         }
150 
151         String cmt = new String("# " + allSphs.sphs.size() + " Hipparcos objects within " + lyLim + " light years") ;
152 
153         allSphs.writeVrml(ofname,new String("hip_main.dat"),cmt,tagVerb,tagCntr,2.5*lyLim) ;
154     }
155 
156 } /* hip2wrl */
```

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

```
1  package de.mpg.mpia.hip ;
2 
3  import java.lang.* ;
4  import java.util.* ;
5 
6  /** A Sphere of fixed radius and center point.
7  * @author Richard J. Mathar
8  * @since 2011-07-31
9  */
10 public class Sphere extends Point3D  {
11     /** Radius
12     */
13     public double radius ;
```

```
14
15      /** RGB color values in the range 0 to 1
16      */
17      float[] rgb ;
18
19      /** RGB color values reduced via magnitudes.
20      * Used to describe emittances.
21      */
22      float[] rgbEmit ;
23
24      String tag ;
25
26      /** Ctor given the center and radius.
27      * @param centr The coordinates of the center
28      * @param R The radius.
29      */
30      public Sphere(final Point3D centr, double R)
31      {
32          super(centr.coo[0],centr.coo[1],centr.coo[2]) ;
33          radius = R ;
34          rgb = new float[3] ;
35          rgbEmit = new float[3] ;
36          rgb[0] =rgb[1] =rgb[2] = 0 ;
37          rgbEmit[0] =rgbEmit[1] =rgbEmit[2] = 0 ;
38          tag = new String() ;
39      } /* Sphere */
40
41      /** Ctor given the center and radius.
42      * @param centr The coordinates of the center
43      * @param R The radius.
44      */
45      public Sphere(final double[] centr, double R)
46      {
47          super(centr) ;
48          radius = R ;
49          rgb = new float[3] ;
50          rgbEmit = new float[3] ;
51          rgb[0] =rgb[1] =rgb[2] = 0 ;
52          rgbEmit[0] =rgbEmit[1] =rgbEmit[2] = 0 ;
53          tag = new String() ;
54      } /* Sphere */
55
56      /** Ctor given the center and radius.
57      * @param centrx The Cartesian x coordinate of the center
58      * @param centry The y coordinate of the center
59      * @param centrz The z coordinate of the center
60      * @param R The radius.
61      */
62      public Sphere(final double centrx, final double centry, final double centrz, double R)
63      {
64          super(centrx,centry,centrz) ;
65          radius = R ;
66          rgb = new float[3] ;
67          rgbEmit = new float[3] ;
68          rgb[0] =rgb[1] =rgb[2] = 0 ;
69          rgbEmit[0] =rgbEmit[1] =rgbEmit[2] = 0 ;
70          tag = new String() ;
71      } /* Sphere */
72
73      /**
74      * @since 2011-08-02
75      * @return Representation as x y z r .
76      */
77      public String toString()
```

```
78      {
79          return (super.toString() + " r " + radius );
80      }
81
82  } /* Sphere */
```

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

```
1   package de.mpg.mpia.hip ;
2
3   import java.util.* ;
4   import java.text.* ;
5
6   /** A point in 3D.
7   * @author Richard J. Mathar
8   * @since 2011-05-12
9   */
10  public class Point3D {
11      /** coo[0..2] are the Cartesian coordinates
12      */
13      public double[] coo ;
14
15      /** Ctor for a point at the origin of coordinates
16      * @since 2011-09-20
17      */
18      public Point3D()
19      {
20          coo = new double[3] ;
21          coo[0] = coo[1] = coo[2] = 0. ;
22      }
23
24      /** Ctor with the three cartesian coordinates.
25      * @param x The cartesian x coordinate.
26      * @param y The cartesian y coordinate.
27      * @param z The cartesian z coordinate.
28      */
29      public Point3D(double x, double y, double z)
30      {
31          coo = new double[3] ;
32          coo[0] = x ;
33          coo[1] = y ;
34          coo[2] = z ;
35      }
36
37      /** Ctor with the three cartesian coordinates.
38      * @param x The cartesian x coordinate.
39      * @param y The cartesian y coordinate.
40      * @param z The cartesian z coordinate.
41      */
42      public Point3D(final Double x, final Double y, final Double z)
43      {
44          this(x.doubleValue(), y.doubleValue(), z.doubleValue() ) ;
45      }
46
47      /** Ctor with the three cartesian coordinates.
48      * @param x The cartesian x coordinate.
49      * @param y The cartesian y coordinate.
50      * @param z The cartesian z coordinate.
51      */
52      public Point3D(float x, float y, float z)
53      {
54          coo = new double[3] ;
```

```
55          coo[0] = x ;
56          coo[1] = y ;
57          coo[2] = z ;
58      }
59
60      /** Ctor from a vector in 3D
61      * @param xyz The vector ofthe three cartesian coordinates.
62      */
63      public Point3D(final double[] xyz)
64      {
65          coo = new double[3] ;
66          coo[0] = xyz[0] ;
67          coo[1] = xyz[1] ;
68          coo[2] = xyz[2] ;
69      }
70
71      /** Ctor from a vector in 3D
72      * @param xyz The vector ofthe three cartesian coordinates.
73      */
74      public Point3D(final float[] xyz)
75      {
76          coo = new double[3] ;
77          coo[0] = xyz[0] ;
78          coo[1] = xyz[1] ;
79          coo[2] = xyz[2] ;
80      }
81
82      /** Ctor with the a tilt angle and an azimuthal direction.
83      * The result is a vector of length unity.
84      * @param z Zenith (tilt) angle or direction (rad)
85      * @param A Azimuth angle or direction (rad)
86      */
87      public Point3D(double z, double A)
88      {
89          coo = new double[3] ;
90          coo[0] = Math.sin(z)*Math.cos(A) ;
91          coo[1] = Math.sin(z)*Math.sin(A) ;
92          coo[2] = Math.cos(z) ;
93      }
94
95      /** Deep copy.
96      */
97      protected Point3D clone()
98      {
99          return new Point3D(coo) ;
100     }
101
102     /** Return a copy, normalized to unit length
103     * @return a shrunk or stretched copy with length 1.
104     * @since 2011-06-15
105     */
106     public Point3D normalize()
107     {
108         final double l = len() ;
109         return new Point3D(coo[0]/l, coo[1]/l, coo[2]/l ) ;
110     }
111
112     /** Difference vector to another point.
113     * @param oth The other point at the tail of the difference vector.
114     * @return diff where this + diff = oth in the vectorial sense.
115     */
116     public Point3D to(final Point3D oth)
117     {
118         return new Point3D( oth.coo[0]-coo[0], oth.coo[1]-coo[1], oth.coo[2]-coo[2]) ;
```

```
119     }
120
121     /** Difference vector to another point.
122     * @param oth The other point at the tail of the difference vector.
123     * @param t The (signed) stretch parameter for oth.
124     * @return this - t* oth.
125     */
126     public Point3D subtract(final Point3D oth,final double t)
127     {
128         return new Point3D( coo[0]-t*oth.coo[0], coo[1] -t*oth.coo[1], coo[2] - t*oth.coo[2]) ;
129     }
130
131     /** Difference vector to another point.
132     * @param oth The other point at the tail of the difference vector.
133     * @return this - oth.
134     */
135     public Point3D subtract(final Point3D oth)
136     {
137         /* somthing avoiding the implicit call (and probably faster
138         */
139         return new Point3D( coo[0]-oth.coo[0], coo[1] -oth.coo[1], coo[2] - oth.coo[2]) ;
140     }
141
142
143     /** Distance to another point.
144     * @param oth The other point to be measured in distance.
145     * @return The square root of the sum of squares of the Cartesian components.
146     * @since 2011-08-02
147     */
148     public double distance(final Point3D oth)
149     {
150         return subtract(oth).len() ;
151     }
152
153
154     /** Length, considering the components as a vector.
155     * @return The square root of the sum of squares of the Cartesian components.
156     */
157     public double len()
158     {
159         /* perhaps faster than
160         * return Math.hypot(Math.hypot(coo[0],coo[1]),coo[2] );
161         */
162         return Math.sqrt(coo[0]*coo[0] + coo[1]*coo[1] + coo[2]*coo[2] );
163     }
164
165
166     /** blank separated string.
167     * @return Three blank-separated numbers of the 3 Cartesian coordinates.
168     */
169     public String toString()
170     {
171         return new String(coo[0]+ " " + coo[1] + " " + coo[2]) ;
172     }
173
174
175     /** Construct a rotation and angle that moves  the point perpendicular to the direction of its direction.
176     * @return A vector  with components [0,1,2] meanign the rotation axis and [3] the rotation angle (radians).
177     * @since 2015-07-31
178     */
179     public double[] perp()
180     {
181
182         /* Given the coo[0,1,2] with x = r cos phi sin theta, y = r sin phi sin theta and z=r cos theta,
```

```
183         * (theta polar angle, phi azimuth in the spherical coordinates) we construct the
184         * orthogonal matrix that rotates z = (0,0,1) into ( cos phi sin(theta-90), sin(phi) sin(theta-90),cos(theta-90)
185         * = ( -cos phi cos theta, -sin(phi) cos theta, sin theta)
186         * and y=(0,1,0) into
187         * (cos phi sin theta, sin phi sin thea, cos theta). This is equivalent to constructing
188         * a vector that is orthogonal to the original (x,y,z) vector.
189         * Of course this sends the cross product x=y X z into the [cos phi cos theta, sin phi  cos theta , -sin theta ]
190         * This defines the three columns of the rotation matrix.
191         * (sp     cp st    -cp st)
192         * (-cp    sp st    -sp st)
193         * (0       ct      st)
194         * Match this with the rotation
195         * matrix of Appendix A in arxiv:1410.1885 by computing the direction of the rotation
196         * axis (which has eigenvalue 1). The trace of the matrix is 1+2*cos(rotation angle)
197         * = sin(phi)+sin(phi)*sin(theta)+sin(theta)
198         */
199
200         /* y/x = sin(phi)/cos(phi)
201         */
202         final double phi = Math.atan2(coo[1],coo[0]) ;
203         final double cphi = Math.cos(phi) ;
204         final double sphi = Math.sin(phi) ;
205         /* sqrt(x^2+y^2) = r*sin(theta) >0
206         */
207         final double xys = Math.hypot(coo[0],coo[1]) ;
208         /* sqrt(x^2+y^2)/z = sin(theta)/cos(theta)
209         */
210         final double theta = Math.atan2(xys,coo[2]) ;
211         final double ctheta = Math.cos(theta) ;
212         final double stheta = Math.sin(theta) ;
213
214         double[] rax = new double[4] ;
215         /* rotation angle (radians)
216         */
217         rax[3] = Math.acos( (sphi+sphi*stheta+stheta -1.0)/2.0 );
218
219         /* sine of the rotation angle
220         */
221         final double srot = Math.sin(rax[3]) ;
222
223         if ( srot == 0. || len() < 1.e-13)
224         {
225             /* avoid degenerate division through 0 further down
226             */
227             rax[0] = rax[1] =0.0 ;
228             rax[2] = 1.0 ;
229             rax[3] = 0.0 ;
230         }
231         else
232         {
233             /*  Omega[3,2]-Omega[2,3] = ct +sp*st = 2*sin(rot angle)*first component
234             */
235             rax[0] = (ctheta + sphi*stheta)/(2.*srot) ;
236
237             /*  Omega[1,3]-Omega[3,1] = -cp*st = 2*sin(rot angle)*second component
238             */
239             rax[1] = -cphi*stheta/(2.*srot) ;
240
241             /*  Omega[2,1]-Omega[1,2] = -cp-cp*st = -cp(1+st) = 2*sin(rot angle)*third component
242             */
243             rax[2] = -cphi*(1.0+stheta)/(2.*srot) ;
244         }
245
246         return rax ;
```

```
247     } /* perp */
248
249 } /* Point3D */
```

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

```
1   package de.mpg.mpia.hip ;
2
3   import java.io.* ;
4   import java.util.* ;
5   import java.text.* ;
6
7   /** A collection of elements of type Sphere.
8   * @author Richard J. Mathar
9   * @since 2012-02-18
10  */
11  public class SphereSet {
12
13      /** The individual spheres
14      */
15      public Vector<Sphere> sphs ;
16
17      /** end-of-line separator in files
18      */
19      static String eol = System.getProperty("line.separator") ;
20
21      /** Ctor for an empty set of spheres.
22      */
23      public SphereSet()
24      {
25          sphs = new Vector<Sphere>() ;
26      } /* SphereSet */
27
28      /** Ctor bundling a set of existing spheres.
29      * @param spheres The vector of spheres to be collected.
30      */
31      public SphereSet(Vector<Sphere> spheres)
32      {
33          sphs = spheres ;
34      } /* SphereSet */
35
36      /** Add a sphere.
37      * @param s The sphere to be added
38      */
39      public void add(final Sphere s)
40      {
41          sphs.add(s) ;
42      } /* add */
43
44      /** Create one VRML97 file which shows the spheres.
45      * @param ofname Name of the output file
46      * @param sname Name of the associated assembly/part
47      * @param cmt Additional comment lines
48      * @param tagverb 0: no tags, 1: only standard names 2:all HIP numbers
49      * @param tagCntr 0=billboard, 1=fixed, 2=facing
50      * @param position Distance of the default view point from the center
51      * @since 2011-08-25
52      */
53      public void writeVrml(String ofname, String sname, String cmt, final int tagverb, final int tagCntr, final double p
54      {
55          try
56          {
```

```
57              StringWriter v = new StringWriter() ;
58              BufferedWriter b = new BufferedWriter(v) ;
59              final String eol = System.getProperty("line.separator") ;
60
61              b.write( vrmlHdr(eol,sname) );
62
63              b.write(cmt);
64              b.newLine() ;
65
66              /* Viewpoint { position x y z orientation ax ay az angl fieldOfView 0.sth description "Camra 1" }
67              */
68              b.write("Viewpoint { position 0 0 " + position + " orientation 0 0 1 0.0 }");
69              b.newLine() ;
70
71              if ( tagCntr ==0 && tagverb > 0 )
72              {
73                  for(int c=0 ; c < sphs.size() ; c++)
74                  {
75                      Sphere thisp = sphs.elementAt(c) ;
76                      /* print the tag if either tagverb >=2 or if it does not start with one
77                      * of the boring HD or HIP numbers.
78                      */
79                      boolean doPrint = ( tagverb > 1 )
80                          || ( ! thisp.tag.startsWith("HIP") && ! thisp.tag.startsWith("HD") ) ;
81
82                      if (thisp.tag.length() == 0 )
83                          /* optimization of output: empty tag, nothing to print on the billboard */
84                          doPrint = false ;
85
86                      if ( ! doPrint)
87                          continue ;
88
89                      b.write("Transform { ") ;
90                      b.newLine() ;
91
92                      /* the VRML coordinate system is x=right, y=up and z=towards viewer, so
93                      * we redefine the original coordinate set (x,y,z) -> (x,z,-y).
94                      */
95                      b.write("translation " + thisp.coo[0] + " " + thisp.coo[2] + " " + (-thisp.coo[1]) ) ;
96                      b.newLine() ;
97
98                      b.write(" children [ Billboard { axisOfRotation 0 0 0 children [") ;
99                      b.newLine() ;
100
101                     b.write("\tShape { geometry Text { string [ \"" + thisp.tag +"\" ] length [0.5 0.4 0.0 ] } } " ) ;
102                     b.write("] } ]") ;
103                     b.newLine() ;
104
105                     b.write("}") ;  /* end of Transform */
106                     b.newLine() ;
107                 }
108             }
109
110             b.write("Transform {") ;
111             b.newLine() ;
112             b.write("children") ;
113             b.newLine() ;
114             b.write("[") ;
115             b.newLine() ;
116
117             /* http://gun.teipir.gr/VRML-amgem/spec/part1/nodesRef.html#Shape */
118             if( sphs != null)
119             {
120                 for(int c=0 ; c < sphs.size() ; c++)
```

```
121                 {
122                     Sphere thisp = sphs.elementAt(c) ;
123                     b.write("Transform { ") ;
124                     b.newLine() ;
125                     /* the VRML coordinate system is x=right, y=up and z=towards viewer, so
126                     * we redefine the original coordinate set (x,y,z) -> (x,z,-y).
127                     */
128                     b.write("translation " + thisp.coo[0] + " " + thisp.coo[2] + " " + (-thisp.coo[1]) ) ;
129                     b.newLine() ;
130                     if ( tagCntr == 2)
131                     {
132                         final double[] rang = thisp.perp() ;
133                         /* again the calculation within Point3d.pere() uses standard cartesian
134                         * coordinates which must be redefined in the VRML97 coordinate system.
135                         */
136                         b.write("rotation " + rang[0] + " " + rang[2] + " " + (-rang[1]) + " " + rang[3] ) ;
137                         b.newLine() ;
138                     }
139 
140                     b.write("children [ Shape {") ;
141                     b.newLine() ;
142 
143                     /* shape the center as a box with flat shape along the ecliptic
144                     */
145                     if ( thisp.len() < 1.e-12 )
146                         b.write("\tgeometry Box { size " + (2.*thisp.radius) + " " + thisp.radius/2 + " " + thisp.radiu
147                     else
148                         b.write("\tgeometry Sphere { radius " + thisp.radius + " } ") ;
149                     b.newLine() ;
150                     b.write("\tappearance Appearance { material Material { ") ;
151                     b.write(" emissiveColor " + thisp.rgbEmit[0] + " " + thisp.rgbEmit[1] + " " + thisp.rgbEmit[2] ) ;
152                     b.write(" diffuseColor " + thisp.rgb[0] + " " + thisp.rgb[1] + " " + thisp.rgb[2] ) ;
153                     b.write(" } }") ;
154                     b.write("\t}") ;
155 
156                     b.newLine() ;
157 
158                     if ( tagverb >0 && tagCntr > 0)
159                     {
160                         if ( tagverb > 1)
161                             b.write("\tShape { geometry Text { string [ \"" + thisp.tag +"\" ] length [0.5 0.4 0.0 ] }
162                         else if ( ! thisp.tag.startsWith("HIP") && ! thisp.tag.startsWith("HD") )
163                             b.write("\tShape { geometry Text { string [ \"" + thisp.tag +"\" ] length [0.5 0.4 0.0 ] }
164                     }
165                     else
166                         b.write("\t# " + thisp.tag) ;
167 
168                     b.newLine() ;
169                     b.write("\t] ") ; // end children
170                     b.newLine() ;
171 
172                     b.write("}") ; //end transform
173                     b.newLine() ;
174                 }
175             }
176 
177             b.write("]") ;  /* end of children */
178             b.newLine() ;
179 
180             b.write("}") ; /* end of Transform */
181             b.newLine() ;
182 
183             b.flush() ;
184             {
```

```
185                 try
186                 {
187                     Txt2File.writeTxt(v.toString(),ofname) ;
188                 }
189                 catch ( Exception ex)
190                 {
191                     System.out.println(ofname + " not written") ; /*does not matter*/
192                 }
193             }
194         }
195         catch( Exception ex)
196         {
197             ex.printStackTrace() ;
198         }
199     } /* writeVrml */
200 
201     /** Create VRML header
202     * @param sname The title entry.
203     * @param eol The end-of-line terminator; a line feed in VRML97 and
204     *  not necessarily the end-of-line terminator of the operating system.
205     * @return The header with the title and info lines.
206     * @since 2012-02-18
207     */
208     String vrmlHdr(final String eol, final String sname)
209     {
210         String hdr ;
211         hdr = new String("#VRML V2.0 utf8" + eol) ;
212         hdr += "WorldInfo {"+ eol;
213         hdr += "title \""+  sname+"\"" + eol;
214         hdr += "info [" ;
215         hdr += "\"" + getClass().getName() + "\"," ;
216         hdr += "\"" + (new Date()).toString() + "\", \"" + System.getenv("USERNAME") +"\"";
217         hdr += "]" + eol;
218         hdr += "}"+ eol;
219         return hdr ;
220     } /* vrmlHdr */
221 
222 } /* SphereSet */
```

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

```
1  package de.mpg.mpia.hip ;
2  
3  import java.io.* ;
4  import java.awt.* ;
5  import java.awt.event.* ;
6  import javax.swing.* ;
7  
8  /** An interface which puts a String into a text file or a byte sequence into a binary file.
9  * The strategy is to provide two ways of writing: one supported by static methods
10 * which force the contents into the file, not asking for permission to overwrite,
11 * the other one with a ctor that displays a confirmation dialog if these files exist.
12 * @author Richard J. Mathar
13 * @since 2011-06-09
14 */
15 public class Txt2File {
16 
17 
18 
19 
20     /** Name of the file that will receive the output.
21     */
```

```
22      private String fname ;
23
24      /** True if the files are opened in append mode by default. False if overwrite mode.
25      * In most applications, in particular .stl and .plt files, appending is not wise, so
26      * the default is declared to be false, and the applications ought initiate the
27      * instance of the class with last parameter equal to true to prevent overwriting.
28      */
29      private static boolean PRINT_APPEND = false;
30
31      /** Ctor providing a text and an output file name.
32      * This is the interface to the safe (interactive) mode that asks before overwriting.
33      * @param logTxt What is to be placed into the file
34      * @param appName Name of the application for use in window headers
35      * @param foutname Fully qualified file name targeted for output.
36      */
37      public Txt2File(final String logTxt, final String appName, final String foutname)
38      {
39          try
40          {
41              writeTxt(logTxt,foutname) ;
42          }
43          catch(Exception ex)
44          {
45              System.out.println(ex) ;
46          }
47      }
48
49      /** Place a string into an ASCII file.
50      * Batch mode without any user interaction.
51      * @param logtxt The text to be appended. This text includes any cr and lf's.
52      * @param fName The file to be changed. Overwritten or appended if existing (depending on the
53      *   the configuration variable PRINT_APPEND), created if non-existing.
54      * @throws IOException If the writing did not succeed.
55      * @since 2013-04-15 Create parent directory if not yet existing.
56      */
57      public static void writeTxt(final String logtxt, final String fName) throws IOException
58      {
59          if ( fName.length() > 0 && logtxt.length() > 0)
60          {
61              /* create parent directory if not yet existing
62              */
63              File parDir = new File(fName).getParentFile() ;
64              /* parDir may be null if this name is just the single name and does not
65              * mention its parent.
66              */
67              if ( parDir != null)
68              {
69                  if ( ! parDir.exists() )
70                      parDir.mkdirs() ;
71              }
72
73              FileWriter f = new FileWriter(fName,PRINT_APPEND) ;
74              BufferedWriter r = new BufferedWriter(f) ;
75              r.write(logtxt) ;
76              r.close() ;
77              f.close() ;
78          }
79          } /* writeTxt */
80
81  } /* Txt2File */
```

11. 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
```

12. File hipparcos/tools/ident2.txt

The association of Hipparcos ID's with Henry-Draper numbers [6–8] 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
```

13. 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
```

14. 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.*, Astron. Astrophys. **323**, L49 (1997).
[2] R. Carey and G. Bell, *The Annotated VRML 97 Reference* (Addison Wesley, 1999).
[3] ISO/IEC, *VRML97 Functional specification and VRML97 External Authoring Interface (EAI)* (2006).
[4] T. J. Lukka and J. A. Stewart, *FreeWrl* (2015).
[5] C. J. Fluke, D. G. Barnes, and N. T. Jones, Publ. Astron. Soc. Austr. **26**, 64 (1009).
[6] A. J. Cannon and E. C. Pickering, Ann. Harvard Coll. Obs. **91**, 1 (1918).
[7] A. J. Cannon and E. C. Pickering, Ann. Harvard Coll. Obs. **98**, 1 (1923).
[8] A. J. Cannon and E. C. Pickering, Ann. Harvard Coll. Obs. **99**, 1 (1924).