Visualization of Milky Way satellite galaxies

ViSaGE - Visualized Satellite Galaxies Environment

Bachelorarbeit in Physik

angefertigt am Argelander-Institut für Astronomie
vorgelegt der Mathematisch-Naturwissenschaftlichen Fakultät der Universität Bonn

1. **Gutachter:** Prof. Dr. Pavel Kroupa
2. **Gutachter:** Prof. Dr. Klaas S. de Boer

Autor: Manuel Hahn (manuel.hahn@gmx.de)
Bahnhofstraße 2, 54306 Kordel
Matrikelnummer: 1564249

Bonn, September 2009

Contents

1 Introduction

The theory of structure formation within the framework of cold dark matter (CDM) cosmology makes predictions for Milky-Way-Galaxy-type systems concerning number and shape of the distribution of its satellites.

One prediction of the CDM theory is, that the Milky Way Galaxy (MWG) should contain about 500 sub-haloes with masses $M \geq 10^8 M_\odot$ within 500 kpc (Moore et al. 1999). As Kroupa et al. in 2005 stated, there had been found only 13 satellites within the distance of 500 kpc arround the MWG. In recent years, there have been found several other satellites including a new class of ultra-faint companion galaxies with extremly low stellar densities (see Metz et al. 2009a), but the number of satellites is still an order of magnitude lower than preditced by the CDM theory.

Secondly, the CDM theory makes a prediction concering the shape of the distribution of its satellites. The theoretical sub-structure distribution of MWG-type hosts in the framework of CDM theory should be essentially isotropic (i.e. spherical), what was researched by Kroupa et al. (2005). They tested the spatial distribution of the observed MWG satellites against the null-hypothesis, that it is drawn randomly from a spherical Dark Matter (DM) sub-structure distribution. They found that this hypothesis can be excluded with a confidence of 99.5 per cent and that the anisotropy of the distribution is such that the MWG satellites form a disk-like structure which lies nearly perpendicularly to the plane of the MWG. Metz et al. (2007, 2009a) expanded this research and also considered the meanwhile found satellites. They confirmed the disk-like structure of the MWG satellites, called disc-of-satellites (DoS), which is inclined by about 88° with respect to the MWG disc and which has a root-mean-square height of about 20 kpc (see figure 1).

As stated above, the theoretical number and spatial sub-structure distribution of MWG-type hosts, as predicted by the CDM cosmology, is highly inconsistent with the observed MWG satellites. This problem has become generally knwon as 'missing satellite problem' and several approaches have been made to solve it. One can classify these different approaches to explain the missing satellites in those using CDM theory and those which try to explain the missing satellites by giving other explanations besides CDM theory.

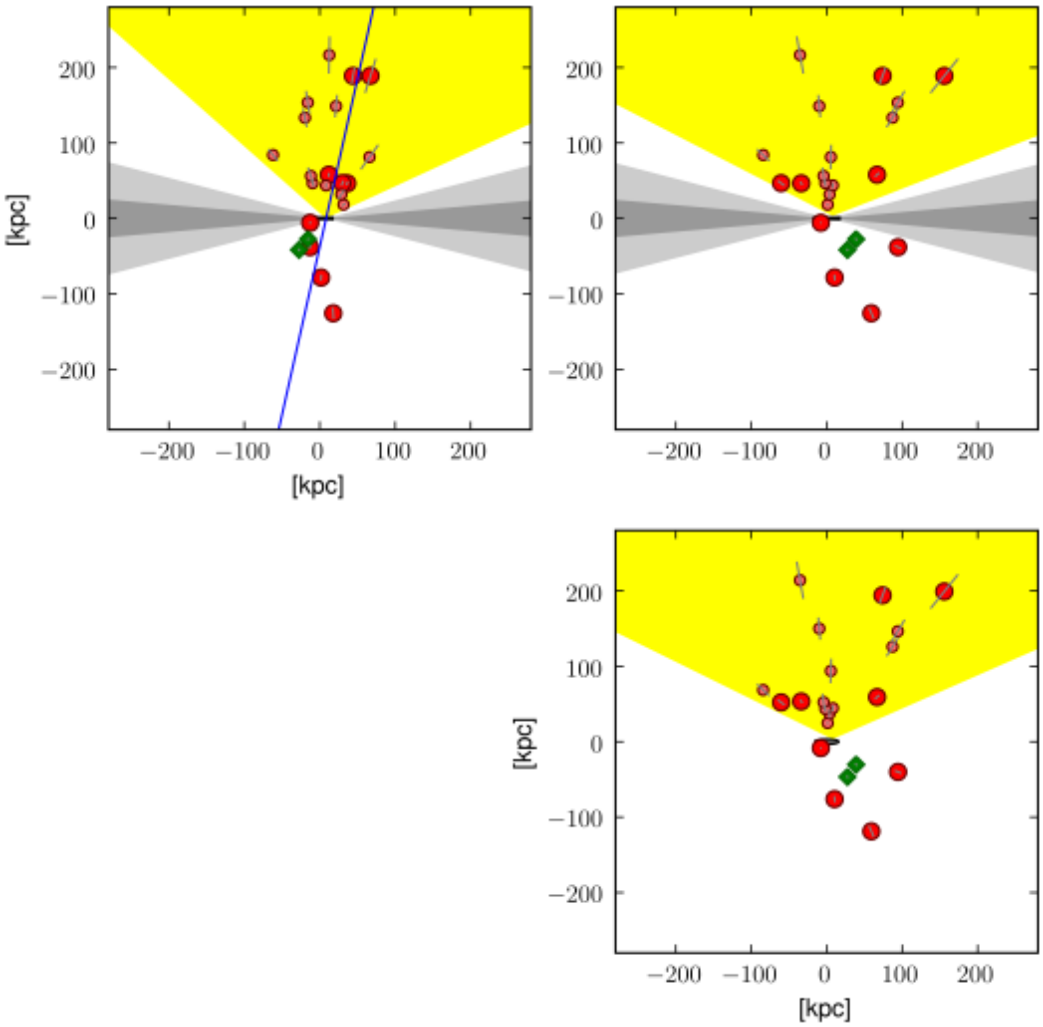

Figure 1: Taken from Metz et al. (2009a): The 3D distribution of the MW satellite galaxies. In the top-left panel an edge-on view onto the fitted DoS as given in MKJ07 is shown, derived using the large circles and diamonds. The MW disc, located in the centre of the plot, is seen edge-on. In the top-right panel, a view rotated by 90° about the polar axis of the MW is shown, and in the lower-right panel a face-on view on to the fitted disc is plotted. The Magellanic Clouds are marked by diamond symbols, the dwarf spheroidals by circles, whereby the smaller circles mark the newly discovered satellites (table 1 from Metz et al. (2009a)). Uncertainties are indicated by light grey sticks. In addition, the obscuration region, $|b| < 5°$, of the MW is shown as the dark-shaded region (the light-shaded region being 15° obscuration region). The projected northern sky coverage region of the SDSS is indicated by the yellow-coloured area.

After Kroupa et el. (2005) mentioned the fact that a correlation between the MWG satellites would naturally arise if they were of tidal origin, Metz & Kroupa (2007) discussed this issue in further detail. The so-called tidal dwarf galaxies (TDG) arise from fundamental physical processes, i.e. conservation of energy and angular momentum, and should be produced in any hierarchical structure formation theory. In 2008, Metz et al. found that the total MWG satellite angular momentum is co-aligned with the normal to the DoS plane, what implies that the DoS is a rotationally supported structure. This is a natural consequence if the MWG satellites are of tidal origin.
An explanation for the highly anisotropic spatial distribution of the MWG satellites, the highly inclined DoS, could be that the MWG satellites entered the MWG halo as a group, what was researched by Metz et al. (2009).

As a reaction to the 'missing satellite problem', the supporter of CDM theory developed CDM simulations, tailored specifically to solve this problem. Metz et al. (2008) compared these simulations with orbital poles and projected uncertainties of MWG satellites, derived from available proper motion measurements. They argued that the sub-haloes produced in these CDM simulations cannot fully describe the entire population of satellites orbiting the MWG.
Furthermore, the supporter of CDM theory argue that the Sloan Digital Sky Survey (SDSS), with which most of the recently discovered satellites were found, has a limted sky coverage area and thus lead to a selection bias (see also figure 1). This was researched by Metz et el. (2009). Including the biasing effects due to the SDSS sky coverage area, their bootstrapping analysis showed that it can be excluded with a confidence of 99.5 per cent, that the MWG satellites are drawn randomly from an isotropic distribution.

As mentioned above, there are several approaches for solving the 'missing satellites problem' which are in parts mutually exclusive. The questions, whether the MWG satellites are or are not DM dominated and what the origin of the MWG satellites is, are from utter importance, especially for modern cosmological theory. It is therefore very important to determine the exact number and positions of the MWG satellites. Hence, special search campaigns like the Stromlo Missing Satellite Survey (Jerjen H. et al., 2009) were initiated and in the next few years, new MWG satellites are expected to be discovered.

The aim of this thesis is to develop a software, that can easily be used to visualize the already knwon MWG satellites and the upcoming newly discovered ones. With the help of this software, it shall be easy to determine the relative position of the new satellites compared to the DoS and generally its position in the galactocentric coordinate system. This software can be used as a 'DoS-quick-test', i.e. a test, whether a newly discovered satellite lies in between the DoS or not.

After having done the scientific research, it is analyzed how the software is supposed to work in order to help people doing their research (Requirement analysis, chapter 2). Afterwards, the technhology research is done in order to find technologies that can be used to realize this software (System design and - specification, chapter 3). After having done some preliminary work which is necessary for coding, the software is finally coded and tested (Implementation and testing, chapter 4). Subsequently, the deployment is discussed (Deployment, chapter 5) before finally some concluding remarks are made (Concluding remarks, chapter 6).

2 Requirement analysis and - specification

2.1 Functional requirements

The software is supposed to visualize the satellites of the MWG in a galactocentric coordinate system. In order to get a 'natural feeling' for the positions of the satellites in space, i.e. a view according to humans binocular vision, the visualization should be three dimensional (3D).

The positions of the upcoming new discovered satellites will be given in equatorial coordinates, i.e. right ascencion and declination, and distance. Therefore the software is supposed to offer an administrative user interface to enter the positions of the new satellites in equatorial coordinates and distance, which will be automatically transformed to galctocentric coordinates.

To properly analyze the satellites and their relation to each other, it is important to be able to filter them. As for example shown in Metz et al. (2007), they used different data sets, morphological subsamples und kinematic subsamples of the satellites. Therefore several so-called 'standard filter' (for a definition see chapter 2.3) should be offered and additionally a possibility to manually select and deselect the satellites. Therewith it is possible to filter the satellites in every possible way.

To research the relative position of the satellites to the DoS, the software is supposed to be able to add a disc, given by a normal vector, a translation vector and a thickness, to the visualization. If one added a new satellite and the DoS to the visualization, one should quickly be able to decide, whether the new satellite is in between the DoS or not. ($\rightarrow$ DoS-quick-test)

2.2 System requirements

The software is supposed to be accessed through the internet. Furthermore, the software should be able to being run from most of the recent operating systems and with most of the recent browsers. The administrative user interface should only be accessible to authorized people. Additionally, the software is supposed to be developed in that way, that its components are as dynamically and customizable as possible. For example should it be possible to just set a configuration flag to change the appearance of the satellites (e.g. color or radius) instead of writing new code. Furthermore, the software should build up on components that are free software or open source.

2.3 Functional specification

The functional specification of the software arises out of the functional requirements in chapter 2.1:

- Administrative user interface to enter the data of the MWG satellites with the following data fields:
  - Name [unique identifier]
  - Equatorial coordinates:
    Right ascension and declination in the epoch J2000,0
  - Distance
  - Luminosity
  - Type (e.g. dE, dSph etc.)
  - Source (e.g. Mateo, M.L., 1998, ARA&A, 36, 435)
- 3D-Visualization of the MWG satellites in a galactocentric coordinate system
  - 'Galactic Center View':
    4π-solid-angle-view, centered on the galactic center
  - 'External View':
    4π-solid-angle-view, as seen from ∞, with Zoom-In/-Out function
  - Information about selected MWG satellites
  - Information about position of Viewing Platform (VP)
  - Standard filter:
    - Show only objects with galaxy type x (morphological filter)
    - Show only objects from data source x
    - Show only objects with a minimum luminosity of $x \cdot 10^6 L_{sun}$

- Manual filter:
  Show / hide single MWG satellites
- Add a disc, given by a normal vector, a translation vector and a thickness, to the 3D-Visualization

3 System design and - specification

3.1 Overview of 3D-Visualization technologies

In this chapter, the currently available technologies, data storages and user interfaces to build a 3D-Visualization are discussed.

3.1.1 Technologies

OpenGL (Open Graphics Library) is an open specification for a 2D and 3D graphics application programming interface (API), that was developed by Silicon Graphics Inc. in 1992. Application developers are free from licensing requirements and OpenGL is a platform independant (cross-platform) and language independent (cross-language) API (see OpenGL_A). Due to its cross-platform character, OpenGL is the leading 3D-specification in industry (see OpenGL_B).

Direct3D is a 2D and 3D graphics API, that was developed by RenderMorphics in 1992 and bought by Microsoft in 1995. Direct3D is a proprietary API, which will only run on Microsoft platforms (see Direct3D).

S2PLOT is 3D plotting library, that was developed by the Centre for Astrophysics & Supercomputing at Swinburne University of Technology. S2PLOT is freely available for non-commercial and educational use, but the programming library and documentation may not be redistributed or sub-licensed in any form without permission from Swinburne University of Technology. The S2PLOT library is based on OpenGL, was written in C and can be used with C, C++ and FORTRAN programs on GNU/Linux and Apple/OSX (no Microsoft OS). S2PLOT content can be embedded in web pages with Flash and in PDF documents using Adobe Acrobat 3D (commercial software) (see Barnes & Fluke, 2008).

JOGL (Java OpenGL) is a library, that allows OpenGL to be used in the object-oriented Java programming language. JOGL was originally developed by Kenneth Bradley Russell and Christopher John Kline, and is currently being developed by the Game Technology Group at Sun Microsystems. JOGL is distributed under BSD license (Berkeley Software Distribution) and is due to its constituents,

Java and OpenGL, cross-platform (see JOGL).

Java 3D is a scene graph-based 3D API for the object-oriented Java programming language. It runs on top of either OpenGL or Direct3D, depending on the users operating system. Java 3D was originally developed by a collaboration of Intel, Silicon Graphics, Apple and Sun in 1997 and is currently developed by Sun Microsystems. It is distributed under the GNU General Public License (GNU GPL) version 2 and is due to its constituents, Java and Direct3D/OpenGL, cross-plattform (see Java3D).

Adobe Acrobat 3D (Acrobat 3D) is a program, that allows one to insert 3D objects into Adobe PDF (PDF) files, which can be viewed with the Adobe Reader (Freeware, cross-platform). It can import 3D objects from more than 50 third-party file formats and converts it into the Universal 3D (U3D) file format (see chapter 3.1.2), which then can be embedded into the PDF file (see Barnes & Fluke 2008). Acrobat 3D belongs to the family of computer programs called Adobe Acrobat, which was developed by Adobe Systems. Adobe Acrobat is a commercial software and costs about 1,000 EUR (Adobe Acrobat 9 Pro Extended, see Acrobat3D_A). It will only run on Microsoft platforms, but if the PDF file once was created on a Microsoft platform with Acrobat 3D, one can deploy the PDF file and PDF is cross-platform. Therefore - in the context of deploying - Acrobat 3D can be considered as cross-platform (see Acrobat3D_B).

movie15 is an extension package for LaTeX and pdfLaTeX, which provides an interface to embed movies, sounds and 3D objects into PDF files, which can be viewed with the Adobe Reader (Freeware, cross-platform). The exclusive file format for embedding 3D objects is the U3D file format (see chapter 3.1.2). movie15 was developed by Alexander Grahn and is distrubted under the LaTeX Project Public License (LPPL). Due to the fact that movie15 is based on LaTeX, it is cross-platform (see movie15).

3.1.2 Data storages

Universal 3D (U3D) is a compressed file format for 3D computer graphics data. It was defined by a consortium called 3D Industry Forum (3DIF) (consisting of e.g. Intel, Boeing, HP, Adobe Systems, Bentley Systems, Right Hemisphere) and was standardized by Ecma International in 2005. 3D objects, stored in U3D, can natively be embedded into PDF files and interactively visualized by Adobe Reader (since version 7) (see U3D).

Extensible 3D Graphics (X3D) is a XML-based file format for 3D computer graphics data. X3D has been definied by the World Wide Web Consortium

(W3C) in 2001 and has been made to become the official standard for 3D graphics in the internet. X3D is the successor to the Virtual Reality Modeling Languange (VRML) and was ratified as an ISO standard in 2004 (see X3D).

Besides special data storages for 3D graphics, one can consider to store the data in proprietary or self-developed file formats like a **text file**.

Alternatively, one can store the data in a **database** like Microsoft SQL or PostgreSQL. Doing so, it is required to operate a database management system on the server.

3.1.3 User interfaces

Adobe Flash (Flash) is a multimedia platform for adding animation and interactivity to web pages. Flash was developed by Macromedia in 1996 and bought by Adobe in 2006. In order to view Flash files in a web browser, one needs to install the Adobe Flash Player, which is cross-platform. Flash files are written with the commercial software Adobe Flash CS4 Professional, which costs about 800 EUR (see Flash).

Java applet is a platform for executing Java bytecode in web pages. Java applets were introduced by Sun Microsystems in 1995. In order to view Java applets in a web browser, one needs to install a Java Virtual Machine (JVM), which is cross-platform. Java applets can be written with several free and open source software development tools (see JavaApplet).

Adobe Reader is a computer programm to display PDF files. Adobe Reader was first published by Adobe Systems in 1993, it is cross-platform and Freeware. PDF files can be written for example with the commercial software Adobe Acrobat or with the free software LaTeX. Especially, if one wants to build a PDF file with an embedded 3D object, one can either use Adobe Acrobat with its 3D component or LaTeX with the movie15 package (see AdobeReader and movie15).

3.2 System specification

Based on the system requirements in chapter 2.2 and the technical research in chapter 3.1, the following constellation of technologies will be used:

- Technology: Java 3D
- Data storage: X3D
- User interface: Java-Applet

Java 3D is cross-platform and a free software. In contrast to OpenGL, Java 3D uses the Java programming language, which has a simpler object model. Additionally, Java 3D uses Direct3D or OpenGL, depending on what is best for the users operating system. I couldn't determine any advantage of S2PLOT over Java 3D and due to the fact that Java 3D has a much bigger community who develop and maintain the software, I decided to use Java 3D.

For the amount of data one can expect in this project (number of satellites $< 10^3$), a database with a database management system would definitely be oversized. Furthermore, a self-developed file format isn't necessary either, because the data storages, especially made for 3D graphics data, are sufficient. I have chosen X3D as file format, because it is the official standard for 3D graphics in the internet. Besides that, X3D is a XML-based format and can thus being read very well.

Finally, using a Java applet is the natural choice if the underlying technology is Java 3D.

4 Implementation and testing

4.1 Preliminary work

4.1.1 Galactocentric coordinate system

The MWG satellites are supposed to be visualized in the Galactocentric coordinate system, i.e. the standard Galactic coordinate system with the Galactic Center (GC) (instead of the sun) as the center of the coordinate system. Due to the fact that the satellite data is entered in equatorial coordinates, one has to transform these coordinates to galactocentric coordinates. This will be done in 3 steps:

First of all, one has to transform the equatorial coordinates to cartesian coordinates. Equatorial coordinates, given as right ascension α and declination δ, plus the distance r were transformed to cartesian coordinates via (see figure 2):

$$\vec{r}_{\text{cartesian}} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r\cos\delta\cos\alpha \\ r\cos\delta\sin\alpha \\ r\sin\delta \end{pmatrix} \tag{4.1}$$

The second step is to transform the cartesian coordinates to the standard cartesian Galactic coordinate system (where the center of the coordinate system is the sun). The distance earth - sun is approximately $16 \cdot 10^{-6}$ ly (light travels from sun to earth in ≈ 8 minutes, thus one gets the distance $d = \frac{8}{365 \cdot 24 \cdot 60}$ ly). As

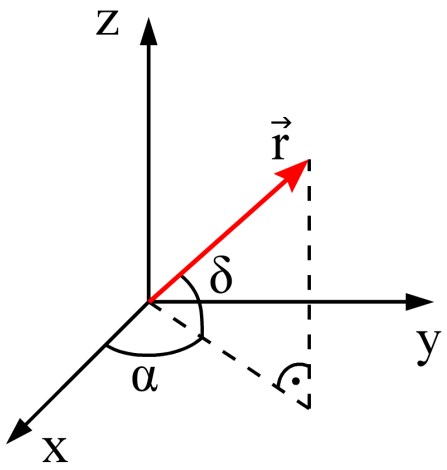

Figure 2: Transformation of equatorial coordinates to cartesian coordinates

the distance earth - sun is very small in comparison to the uncertainties of the distances of the satellites, which are at least 1 kpc = 3,26 ly, one can consider the cartesian coordinates, as calculated in the first step, as heliocentric coordinates. Hence, the transformation from heliocentric cartesian coordinates to standard cartesian galactic coordinates is done by rotating the coordinate system axis by the matrix $\underline{R}$:

$$\vec{r}_{\text{galactic}} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \underline{R} \cdot \vec{r}_{\text{cartesian}} \tag{4.2}$$

The matrix $\underline{R}$ is given by (Manuel Metz, private communication):

$$\underline{R} = \begin{pmatrix} -0.06699 & -0.87276 & -0.48354 \\ +0.49273 & -0.45035 & +0.74458 \\ -0.86760 & -0.18837 & +0.46020 \end{pmatrix} \tag{4.3}$$

In the last step, one has to transform the standard cartesian galactic coordinates to cartesian galactocentric coordinates. This is simply done by subtracting the distance sun - GC, $d_{\text{sun}} := 8.5$ kpc, from the x-coordinate of the standard cartesian galactic coordinate vector:

$$\vec{r}_{\text{galactocentric}} = \begin{pmatrix} x'' \\ y'' \\ z'' \end{pmatrix} = \vec{r}_{\text{galactic}} - \begin{pmatrix} d_{\text{sun}} \\ 0 \\ 0 \end{pmatrix} \tag{4.4}$$

4.1.2 The Java 3D ViewingPlatform

In order to implement Zoom-In/-Out functions and rotation of the 3D objects, one has to modify the Java 3D ViewingPlatform. In Java 3D, there only exists one right-handed coordinate system (z-axis points to the viewer), which is fixed, and the 3D objects are fixed to this coordinate system. The possibility to view the 3D objects in Java 3D is given by the Java 3D ViewingPlatform. One can imagine, that one stands on a viewing platform and watches a nice building. The difference between this viewing platform and the Java 3D ViewingPlatform is, that the latter is able to 'fly' in space. When the 3D world is initialized, the Java 3D ViewingPlatform points to the direction $\vec{d}_{\text{ViewingPlatform}} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$. Thus, if you want to see the GC, you have to shift the Java 3D ViewingPlatform to e.g. $\vec{r}_{\text{ViewingPlatform}} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 50 \end{pmatrix}$. The action performed right now was a Zoom-Out by 50 kpc.

To discuss this in general, one has to imagine an intialized Java 3D ViewingPlatform, which points to the direction $\vec{d}_{\text{ViewingPlatform}} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$ (see figure 3a). One now wants to view the 3D object from the position $\vec{t}$ (see figure 3b). Therefore the Java 3D ViewingPlatform is shifted to this position $\vec{t}$. The problem now is, that the Java 3D ViewingPlatform still points to the direction $\vec{d}_{\text{ViewingPlatform}} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$ and thus one can't see the 3D object. One has to rotate the Java 3D ViewingPlatform in that way, that it points to the origin. This is done by the rotation matrix $\underline{R}$ (see figure 3c).

In the software, the rotation matrix $\underline{R}$ and the translation vector $\vec{t}$ of the current position of the Java 3D ViewingPlatform are given at the bottom of the visualization window (see figure 5).

4.1.3 Description of planes

The software is supposed to be able to add a disc, given by a normal vector $\vec{n}$, a translation vector $\vec{p}$ and a thickness d, to the visualization.

One can mathematically describe a plane by using the Hesse form:

$$\vec{n} \cdot (\vec{x} - \vec{p}) = 0 \tag{4.5}$$

In the software, a 'plane' is described by a cylinder with height d (paramter thickness) and a very big radius ($\approx$ 5,000 kpc; which is one order of magnitude higher than the virial radius of the MWG). The orientation of a cylinder in Java 3D shall be given by the normal vector $\vec{n}'$ of the top surface of the cylinder (see figure 4).

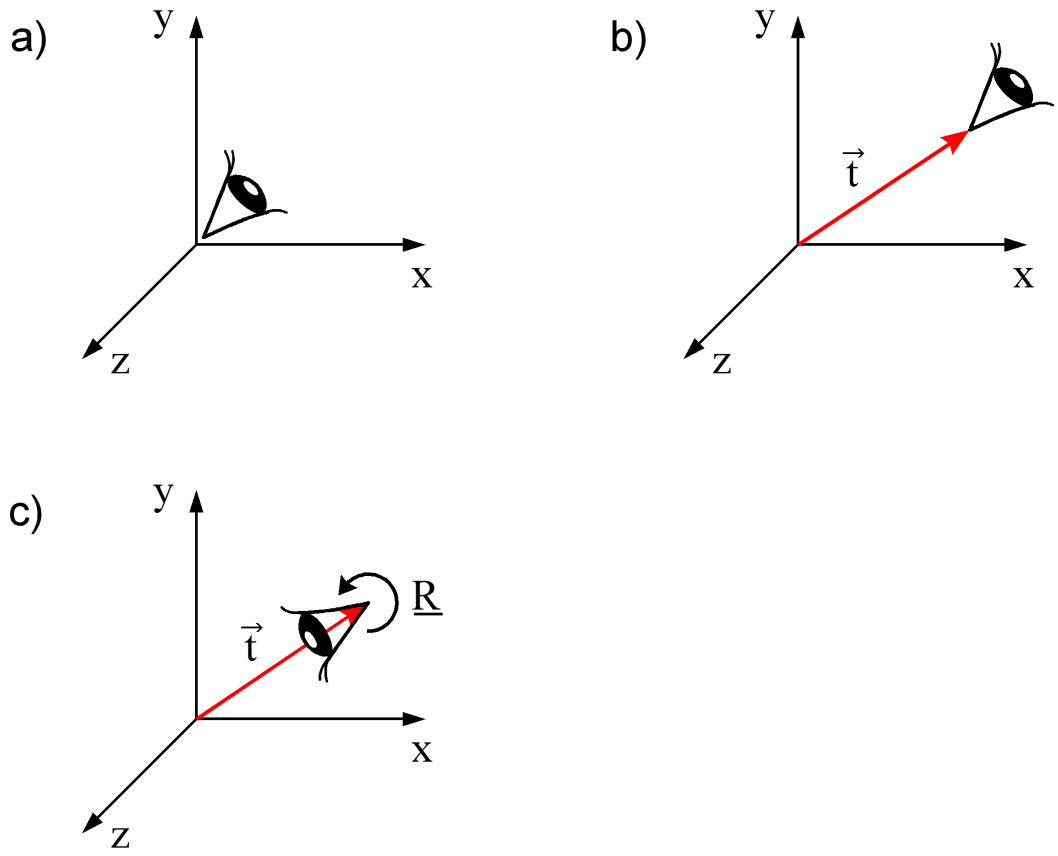

Figure 3: Modifying the Java 3D ViewingPlatform

When initalized, a cylinder has the orientation $\vec{n}' = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ and is positioned at the origin.

First we need to transform the cylinder in that way, that the normal vectors $\vec{n}$ and $\vec{n}'$ point to the same direction. Therefore we have to rotate $\vec{n}'$ arround an axis $\vec{a}$ by an angle ϕ. One gets $\vec{a}$ by building the cross product between $\vec{n}'$ and $\vec{n}$:

$$\vec{a} = \vec{n} \times \vec{n}' \tag{4.6}$$

Now one needs to calculate the angle between $\vec{n}$ and $\vec{n}'$. This is simply given by:

$$\phi = \arccos \left| \frac{\vec{n}' \cdot \vec{n}}{|\vec{n}'| \ |\vec{n}|} \right| \tag{4.7}$$

Finally after rotating the cylinder to the orientation given by $\vec{n}$, one has to shift the cylinder to the position given by the translation vector $\vec{p}$ (see again figure 4).

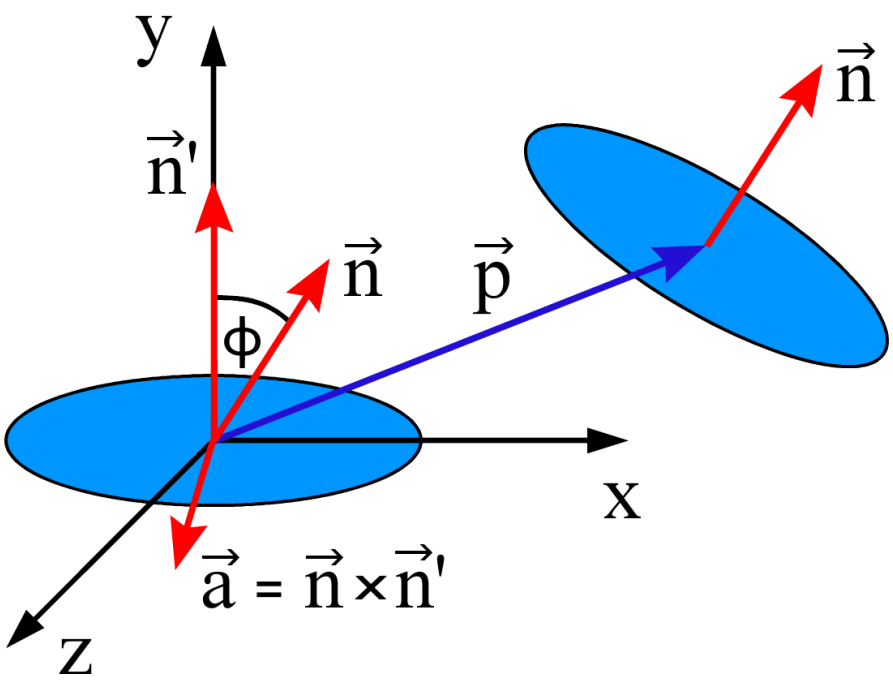

Figure 4: Descripton of a plane in Java 3D

4.1.4 Notes on lights

In order to enhance the 3D feeling in Java 3D, one has to add one or more light sources to the visualization and modify the surface of the 3D objects in that way, that light can be reflected on it. In this software, only one light source is used, i.e. a point source located at the origin (GC), which emits light in all directions. This enhances the 3D feeling in the visualization, as said before, but it has got an other big advantage. If one makes a screenshot of the visualization, for example to publish it, one only gets a 2D image, but one can use the shadings of the surface of the 3D objects to determine the satellites position relative to the origin (GC) (see figure 5).

4.2 Coding

The source code of the software is written in Java with the free, open source and cross-platform software development environment Eclipse (see Eclipse). As already said, the sotfware is written on the base of the 3D API Java 3D. Furthermore, an open source document object model for XML, called JDOM, was used to implement the interaction between Java and the X3D file (see JDOM).

4.3 Testing

During software development, every completed part of the visualization was extensively tested. The provided functions were tested and additionally any constellation of erroneous user inputs were evaluated and tested.

The completed Java applet, embedded in a webpage, was tested on Mac OS X, Windows XP and Linux Ubuntu 9.04 with the browsers Safari, Firefox and Internet Explorer.

5 Deployment

This software is distributed under the name "ViSaGE - Visualized Satellite Galaxies Environment". Due to the fact that Java 3D is licensed under the GNU General Public License (GNU GPL), wich contains a Copyleft condition (see GNUGLP), the software ViSaGE is distributed under the GNU GPL as well.

In the ViSaGE, there has been entered data from 20 MWG satellites. This data was taken from various sources, which are given in the visualization and referenced in this thesis. Several data concerning luminosity and type couldn't be found in these sources, hence if a luminosity (type) for a satellite couldn't be determined, a "0" ("x") was entered. One can manually add / remove / edit satellite data with the ViSaGE administration user interface.

The complete sourde code for the visualization - and administration class as well as the compiled and signed jar file can be found on the enclosed CD. On this CD one can also find a webpage, which contains a welcome page with a general description and an electronic version of this thesis, an installation guide and the administration - and visualization user interface. The webpage can be viewed by executing the index.html which one can find in the root folder of the CD. Finally, on the CD one can find a X3D file named data.x3d, in which the data is stored and some configurations are made.

A screenshot of the ViSaGE can be found in figure 5.

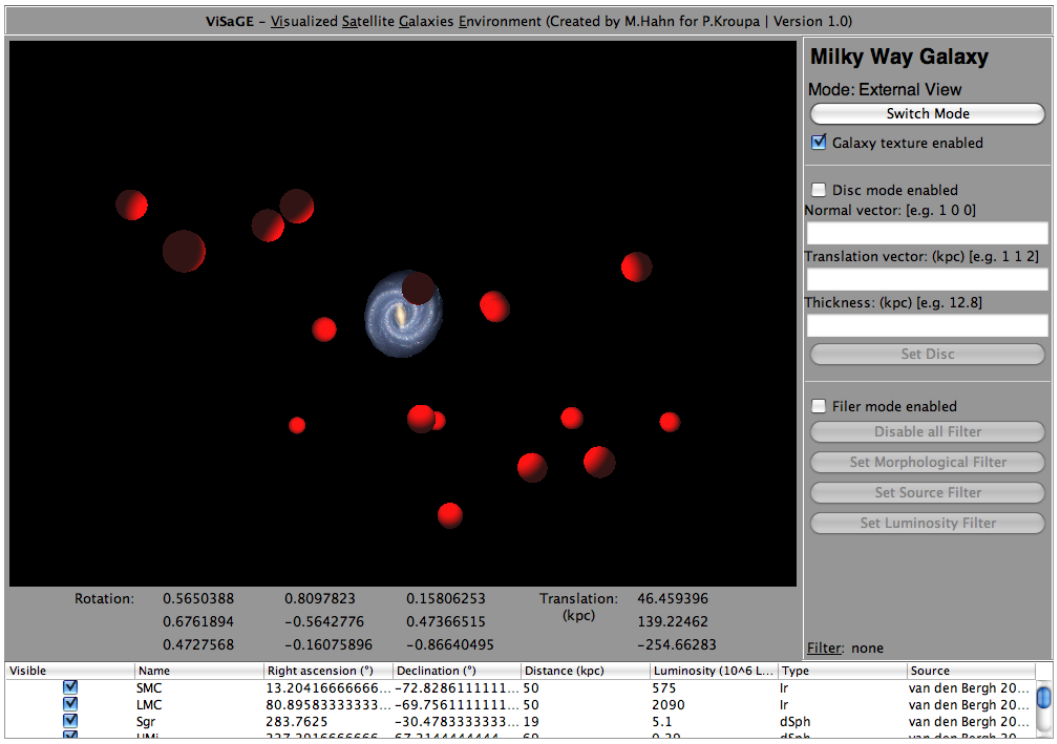

Figure 5: Screenshot of the ViSaGE

6 Concluding remarks

In this bachelor thesis, a Milky Way (MW) satellite galaxies visualization called ViSaGE was developed. With this software, one can add, remove and edit satellites of the MW and visualize them in a 3D environment with rotatation - and Zoom-In / Zoom-Out functions. One further has the possibility to add an arbitrary disc to the visualization and to filter the view of the satellites in various ways. When clicked on a satellite in the visualization, the satellite appears in white color and in the table below one can see further details of the satellite (name, position, distance, luminosity, type and source). It is also possible to select a satellite in the table, what causes the satellite in the visualization to appear in white color. ViSaGE is a Java applet, which can be embedded in a webpage, like it has been done on the webpapge delivered on the enclosed CD.

On the webpage on the enclosed CD one can find the ViSaGE, which displays 20 MW satellite galxies. As one can see in figure 6, the view as described in figure 1, top-left panel, can be reproduced very well. With ViSaGE it should now be easy to visualize newly discovered MW satellite galaxies, which are supposed to be found due to upcoming surveys like the SMSS. And due to this, it should also

be easy to determine, whether a newly discovered satellite lies in between the DoS or not (DoS-quick-test).

To gain a better understanding with respect to the 'missing satellite problem', the distribution of the Andromeda satellites are recently researched as well (see e.g. Metz et al. 2009a). Hence, the ViSaGE should be upgraded in order to display the Andromeda stallites as well. The ViSaGE was designed in that way, that this upgrade can easily be done.

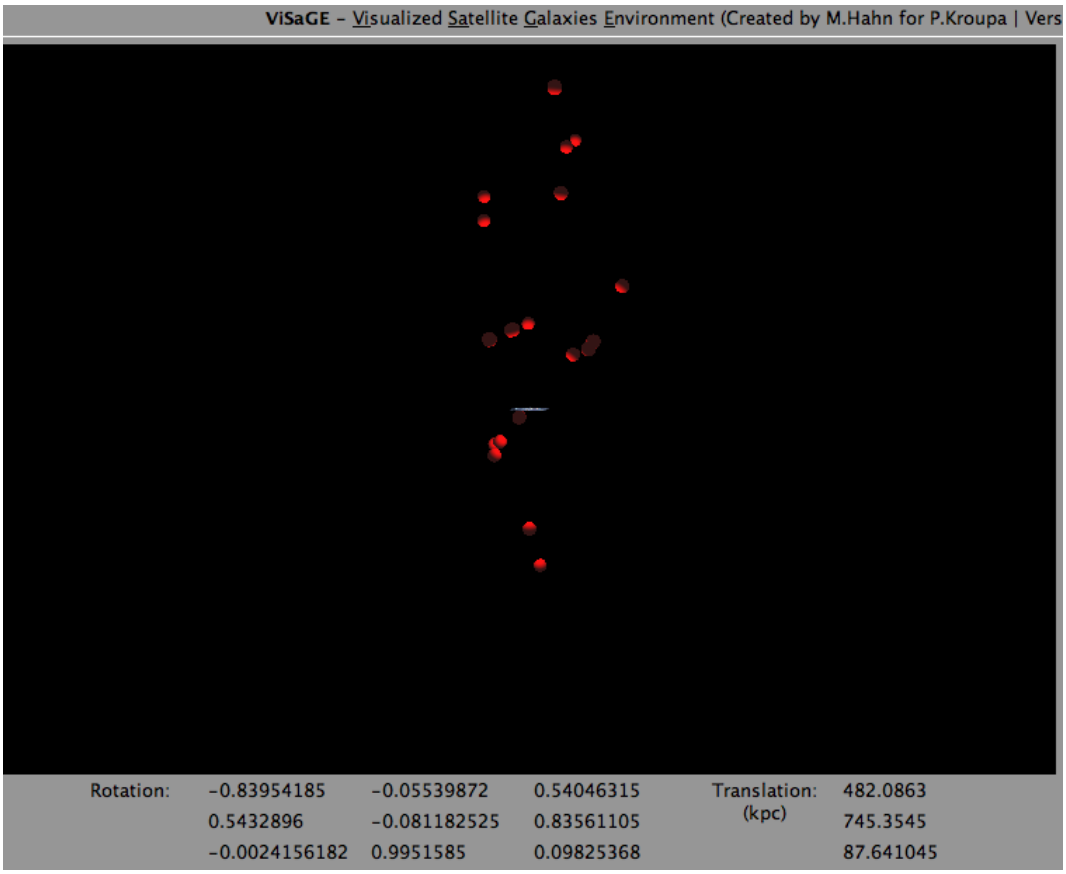

Figure 6: The 3D distribution of 20 MW satellite galaxies in the ViSaGE (see table in ViSaGE for futher details). The MW is shown nearly edge-on. The view described in figure 1, top-left panel, is shown.

References

Acrobat3D_A, as at September 28th, 2009:
https://store.adobe.com/
Acrobat3D_B, as at September 28th, 2009:
http://en.wikipedia.org/wiki/Adobe_Acrobat
AdobeReader, as at September 28th, 2009:
http://de.wikipedia.org/wiki/Adobe_Reader
Barnes, D. G., Fluke, C. J., 2008, NewA, 13, 8, 599
Belokurov, V., et al., 2006b, ApJ, 647, I2, L111
Belokurov, V., et al., 2007, ApJ, 654, 897
Belokurov, V., et al., 2008, ApJ, 686, I2, L83
Direct3D, as at September 28th, 2009:
http://en.wikipedia.org/wiki/Microsoft_Direct3D
Eclipse, as at September 28th, 2009:
http://www.eclipse.org/
Flash, as at September 28th, 2009:
http://en.wikipedia.org/wiki/Adobe_Flash and
https://store.adobe.com/
GNUGPL, as at September 28th, 2009:
http://www.gnu.org/licenses/gpl.html
Java3D, as at September 28th, 2009:
http://en.wikipedia.org/wiki/Java3D
JavaApplet, as at September 28th, 2009:
http://en.wikipedia.org/wiki/Java_applet
JDOM, as at September 28th, 2009:
http://jdom.org/
Jerjen H. et al., as at September 28th, 2009:
http://msowww.anu.edu.au/~jerjen/SMS_Survey.html
JOGL, as at September 28th, 2009:
http://en.wikipedia.org/wiki/Java_OpenGL
Kroupa, P., Theis, C., Boily, C.M., 2005, A&A, 431, 517
Metz, M., Kroupa, P., 2007, MNRAS, 376, 387
Metz, M., Kroupa, P., Jerjen, H., 2007, MNRAS, 374, 1125 (MKJ07)
Metz, M., Kroupa, P., Libeskind N. I., 2008, ApJ 680, 287
Metz, M., Kroupa, P., Jerjen, H., 2009a, MNRAS, 394, 2223
Metz, M., Kroupa, P., Theis, C., Hensler, G., Jerjen, H., 2009b, ApJ, 697, 269
Moore, B., Ghigna, S., Governato, F., et al., 1999, ApJ, 524, L19
movie15, as at September 28th, 2009:
http://www.ctan.org/tex-archive/macros/latex/contrib/movie15/
OpenGL_A, as at September 28th, 2009:
http://www.opengl.org/about/overview/

OpenGL_B, as at September 28th, 2009:
`http://de.wikipedia.org/wiki/OpenGL`
U3D, as at September 28th, 2009:
http://www.ecma-international.org/news/PressReleases/PR_TC43_U3D_Aug05.htm
van den Bergh, S., 2006, AJ, 132, I4, 1571
Walsh, S.M., Jerjen, H., Willman B., 2007, ApJ, 662, I2, L83
Willman, B., et al., 2005, AJ, 129, I6, 2692
X3D, as at September 28th, 2009:
`http://de.wikipedia.org/wiki/X3D` and
`http://www.web3d.org/x3d`
Zucker, D.B., et al., 2006, ApJ, 643, L103

Source of the MWG image, which can be found as a texture inside the visualization:
NASA, JPL-Caltech, R. Hurt (SSC-Caltech), ssc2008-10a
`http://sscws1.ipac.caltech.edu/Imagegallery/image.php?image_name=ssc2008-10a`

List of Figures

Acknowledgments

First of all I want to thank my family and especially my parents for their undivided help and support. I couldn't have finished my bachelor degree without them.

At the University of Bonn I want to thank Pavel Kroupa for his support during writing my thesis and his research group in general for the pleasent working environment. Especially, I want to thank Marcel Pawlowski for answering my questions and testing the software.

Finally, I want to thank Anton Ippendorf for proofreading the thesis and testing the software and Christian Karrasch for digitizing the figures.

This thesis is dedicated to my grandmother Irmgard Piepenbrock-Lieser.

Ich versichere, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate kenntlich gemacht habe.

Bonn, den 29.09.2009 .....

(Manuel Hahn)