

A Planetarium on the Web

Department of Computing
Imperial College
London

Pretesh J Mistry

pjm02@doc.ic.ac.uk

Project Supervisor: *Steffen van Bakel*
Second Marker: *Ian Hodkinson*

June 2005

completed as part of the Beng degree for computing at Imperial College

Acknowledgements

I would like to express my thanks to my supervisor Steffen van Bakel, for spending much time, providing invaluable help and directing me during our project meetings.

I would also like to thank Ian Hodkinson for his constructive comments and suggestions.

Contents

CONTENTS	2
1. ABSTRACT	4
2. INTRODUCTION	5
GOAL	5
REPORT STRUCTURE	5
3. BACKGROUND	7
3.1 POSITIONAL ASTRONOMY AND ASTROMETRY	7
3.1.1 TIME	7
JULIAN DATE	7
UNIVERSAL TIME/GREENWICH MEAN TIME.....	8
SIDEREAL TIME	8
3.1.2 OBSERVER POSITION	10
3.1.3. COORDINATE SYSTEMS	11
HORIZON COORDINATES	11
EQUATORIAL COORDINATES	12
ECLIPTIC COORDINATES	13
COORDINATE CONVERSIONS.....	14
3.1.4 THE STARS	15
VISUAL APPEARANCE	15
3.1.5 THE SUN	16
ORBITS	16
THE SUN'S APPARENT ORBIT	17
CALCULATING SUN'S POSITION	17
SUN'S ANGULAR SIZE	18
3.1.6 THE PLANETS	19
LIMITATIONS OF METHOD:	21
3.1.7 THE MOON	22
ACCURACY.....	23
3.1.8 PROJECTIONS	24
MERCATOR PROJECTION.....	24
POLAR PROJECTION.....	24
ZENITH PROJECTION	24
3.2 EXISTING SOLUTIONS	26
NEAVE'S PLANETARIUM.....	26
SKY VIEW CAFÉ.....	27
YOURSKY	28
3.2.1 EXISTING SOLUTIONS SUMMARY	28
4. DECISIONS	30
4.1 TECHNOLOGY	30
4.1.1 TECHNOLOGY USED	30
PHP – Hypertext Pre-processor [T1].....	30
Portable Network Graphic (PNG) [T2]	30
PostgreSQL [T3].....	30
4.1.2 OTHER POSSIBLE TECHNOLOGIES	31
Java Applet	31
Tomcat & Java Servlet.....	31
4.1.3 TECHNOLOGY SUMMARY	31
4.2 SYSTEM OBJECTIVES	32
4.2.1 SYSTEM SPECIFICATION.....	32
4.2.3 STAR CATALOGUES	33
5. DESIGN	36
5.1 SYSTEM OVERVIEW	36

5.2 DATABASE	36
5.2.1 TABLES & STRUCTURE	36
5.2.2 CATALOGUE PARSING.....	37
5.3 CLIENT-SIDE FRONT-END	38
5.3.1 INTERFACE GENERATION	38
5.3.2 ZOOMING AND PANNING CONTROLS	39
5.4 SERVER-SIDE SCRIPTS	40
6. IMPLEMENTATION	41
6.1 MAINTAINING STATE	41
6.2 GD GRAPHICS LIBRARY	41
6.2.1 BASIC DRAWING FUNCTIONS	41
6.2.2 FINAL GRAPHICS ENGINE	42
6.3 ALGORITHM ISSUES	43
6.3.1 INVERSE TAN FUNCTIONS	43
6.3.2 TESTING	43
6.4 ZOOMING AND PANNING	43
6.4.1 ZOOMING SOLUTION	43
6.4.2 PANNING SOLUTION	44
7. EVALUATION	46
7.1 CORRECTNESS/COMPARISONS	46
Results:	46
7.1.1 ANALYSIS.....	48
Constellations	48
Stars.....	48
Solar System objects	48
Overall Clarity.....	48
Performance	48
8. CONCLUSION	49
8.1 FUTURE WORK	49
8.2 SUMMARY	50
11. BIBLIOGRAPHY	51
11.1 BACKGROUND	51
11.2 TECHNICAL	51
A1. APPENDIX	53
A1.1 SUN'S APPARENT ORBIT AT EPOCH	53
A1.2 ELEMENTS OF THE MOON'S ORBIT AT EPOCH	53
A1.2 ELEMENTS OF THE PLANETARY ORBITS	0

1. Abstract

This report covers the details behind the design and implementation of a web based planetarium. The final product being a web site from which users can generate images of the sky and the celestial objects contained within that portion of the sky.

Much of the research work involved reading through positional astronomy texts and gathering models to help create a simulation of the solar system and the stars that lie beyond it.

The report aims to provide a reasoning of decisions taken and justification of the implementation.

It concludes with an evaluation of the implementation and suggestion of possible future work.

2. Introduction

The sky above us is enormous and filled with millions of objects, some of which are perfectly visible with the naked eye. However with most of us living in urban areas the night sky is becoming increasingly difficult these past few decades to view comfortably. As light pollution continues to crowd the sky it is very rarely that one can recognise constellations and track movements of planets within our own solar system. Nowadays it is only possible to see the brighter stars with the naked eye. [B1]

Stargazing has traditionally involved viewing the night sky with the aid of telescopes, assuming clear conditions in the atmosphere.

The last few years has seen the creation of software to aid stargazers by providing star maps and predictions of positions of objects in the sky as a reference to work from. However these solutions have traditionally been standalone applications, usually requiring the user to download an accompanying star catalogue that may be quite large in size depending on the catalogue and its use.

A natural progression from this has been providing similar features from the standalone application in an online environment via a virtual planetarium. The advantages being that the user, with no requirements, can generate images of the sky through an Internet browser.

This project aims to develop a web application that can be accessed by anyone who would like to view a clear image of the sky, as it would be seen in real life. The user will be able to control what portion of the sky he/she would like to see and a time/date at which that sky should represent.

Goal

Produce a product, in which a user can access a site and generate views of the sky that they would see from the earth's surface with minimum actions by the user. Ensure fast performance and make it available on a wide range of system platforms as possible.

Report Structure

The report is structured according the tasks tackled, as follows:

3. Background – Covers the astronomical models researched, which are to be used in the implementation.

4. Decisions – highlights key choices made regarding the technology used and objectives set out.

5. Design – discusses how the objectives are completed

6. Implementation - How the astronomical models and design was assembled together to form the final product.

7. Evaluation – Analysis of system's result and comparisons with

existing solutions.

8. Conclusion– Suggestions of possible future work

3. Background

The majority of the background involves studying the simulation models required to construct a virtual planetarium. The key topics including, positional astronomy and map projections. The next part will detail existing software solution to creating a planetarium on the web.

3.1 Positional Astronomy and Astrometry

This section deals with all the elements needed in the positioning of objects in reference to the Earth. This subject has much depth and only the fundamental positioning models are required for the purpose of this project.

3.1.1 Time

Time plays an important role for astronomers, and there is a large range of time systems available. However for the positioning of object we only need to concern ourselves with a few.

Julian Date

The most common format for time that we are all accustomed would be splitting it into year, months, date, hours, minutes and seconds. Julian date aims to overcome this cumbersome approach and consolidate all the fields into a single variable. Julian date is defined as *the number of fractional days since a fundamental epoch*¹. An important point to note is that a Julian day is half a step out of time with normal civilian time, i.e. each new Julian day begins at midday. By representing time within a single value the difference between two dates can be easily be found by simply subtracting the Julian dates.

The Julian date of any day is calculated as follows [B14 p7]:

1. set y =year, m =month, d = day
2. if $m=1$ or $m=2$, subtract 1 from y and add 12 to m . Otherwise $y'=y$ and $m'=m$.
3. if the date is later than or equal to 1582 October 15 calculate:
 $A = \text{INT}(y'/100)$ ²
 $B = 2 - A + \text{INT}(A/4)$
Otherwise $B = 0$
4. if y' is negative, calculate
 $C = \text{INT}((365.25 * y') - 0.75)$
otherwise $C = \text{INT}(365.25 * y')$
5. calculate $D = \text{INT}(30.6001 * (m' + 1))$
6. find Julian Date, $JD = B + C + D + d + 1720944.5$

¹ Astronomers have chosen Greenwich mean noon of January 1st 4713 B.C as fundamental epoch.

² INT(N) represents the integer part of the number N.

Universal Time/Greenwich Mean Time

The civilian time that we all use within our daily life's is based on the Earth's rotation. Universal Time (UT) or Greenwich Mean Time (GMT) is derived from the motion of the Sun as observed from the Greenwich meridian at longitude 0°. Using UT/GMT provides the advantage of not having to worry about the observer's position and thus provides a time, which is valid at any location on the Earth.

UT/GMT is calculated as follows [B14 p13]:

1. *convert local civil time to zone time, if necessary correct for daylight saving.*³
2. *subtract the zone correction (i.e. subtract the value of the time zone of the observer)*
3. *if the answer is greater than 24, subtract 24.*

The main use for UT is during the calculations of sidereal time.

Sidereal Time

While UT is based on the Sun's motion, astronomers need a reliable time scale in relation in the stars, such that after 24 hours any given star returns to its same position in the sky. This type of clock is known as the sidereal clock, and its time referred to as the *sidereal time* (ST).

An important point to note is that ST and UT are not the same. This is due to the fact that the Earth rotates around the sun, causing its position to be displaced slightly in the sky after 24 hours have elapsed. This results in a sidereal day being shorter than a solar day (24 hours elapsing in UT).

ST is calculated as follows from a date and UT [B14 p17]:

1. *find the Julian date (JD) corresponding to 0h on this calendar date.*
2. *calculate $S = JD - 2451545.0$*
3. *calculate $T = S/36525.0$*
4. *find $T_0 = 6.697374558 + (2400.051336 * T) + (0.000025862 * T^2)$*
5. *convert UT to decimal hours*
6. *multiply UT by 1.002737909*
7. *add this to T_0 and reduce to the range 0-24 if necessary, which gives Greenwich sidereal time (GST) in decimal hours*

The method above gives the ST at the Greenwich meridian at longitude 0°. However the local sidereal time (LST) would vary depending on the observer's location, more specifically their longitude. The correction to obtain LST is as follows [B14 p20]:

1. *convert GST to decimal hours if necessary*
2. *convert longitude difference in degrees to difference in time by dividing by 15*
3. *if the longitude is W , subtract the difference from GST*

³ Some countries, e.g. Britain add one hour to their local time during the summer months to allow working days to fall within sunlight. This is known as British summer time.

if the longitude is E, add the difference to GST

4. *if the result is greater than 24, subtract 24
if the result is negative, add 24
this result give the LST.*

3.1.2 Observer Position

The most common and longest used method for fixing an observer's position on the Earth's surface has been via the use of latitude and longitude lines. The ancient Greek geographer Ptolemy, who created the grid system and listed the coordinates for place throughout the known world, first devised an early version of this system. It was not until the Middle Ages that further improvements had been made, to use degrees ($^{\circ}$) as the unit of measure. [B16]

To precisely locate a point on the Earth's surface, two values are required; the *latitude* and the *longitude* (figure 1):

Latitudes are horizontal lines running across the Earth's surface in parallel to the equator. In the northern hemisphere the values vary from 0° at the equator to 90° at the North Pole. Likewise for the latitude in the southern hemisphere values vary from 0° to 90° . To help differentiate between the hemispheres being referred to, the latitude usually has an associated letter 'N' or 'S' to indicate the hemisphere.

Longitudes are similar to latitudes except they are vertical lines and vary from 0° to 180° . The hemispheres for longitude are split into the east and west, indicated as 'E' or 'W' respectively.

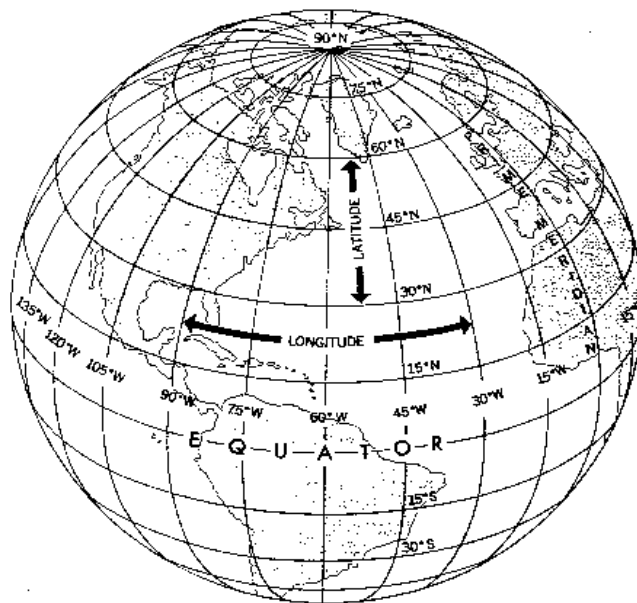


Figure 1: Latitudes and Longitudes of Earth

3.1.3. Coordinate Systems

To enable us fix to a position in the sky we need a frame of reference, which can be provided by the use of a co-ordinate system. A pair of numbers can then be used to uniquely identify each point of the sky.

There are several such coordinate systems available, however for the scope of this project three will be discussed; the horizon system, the equatorial system, and the ecliptic system. These are the systems required in the positioning of the main the celestial bodies under consideration for this planetarium:

- The planets of the Solar system
- The stars visible from the Earth
- The Sun
- The Moon

Horizon Coordinates

This is the most primitive coordinate system, and is based around the observer's horizon. Assume the observer to be located at a point O on the Earth's surface; his horizon would form a circle NESW around him. The hemisphere directly above the observer would then represent half of the celestial sphere on which all the stars would appear to originate from. The point Z, which lies perpendicular to the horizon plane on the celestial sphere, is known as the zenith.

Now imagine a star X on the celestial hemisphere. Its location in relation to the zenith and the horizon could then be determined by two values, *altitude* and *azimuth*. The altitude would represent how high up the star is located from the horizon in degrees and the azimuth would represent how far around the horizon circle from N position (see figure 2).

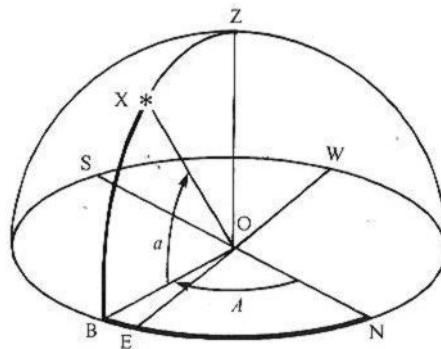


Figure 2: Horizon Coordinates

The primary use for this coordinate system is find the positions of celestial bodies based on an observers location on the Earth at a given time. The altitude/azimuth would both vary for different locations on Earth for any given star at the same time. This problem is overcome by the next coordinate system; Equatorial.

Equatorial Coordinates

This system is based on the plane of the Earth's equator. Assume the observer to be located again at a point O (and within the northern hemisphere), the same horizon circle NESW. Now imagine a new plane defined by the points EYRW, inclined at an angle of $90^\circ - \phi$ to the horizon, where ϕ is the observer's geographical latitude. Therefore 90° to this newly defined plane would the line OP, represent the Earth's rotational axis (see figure 3).

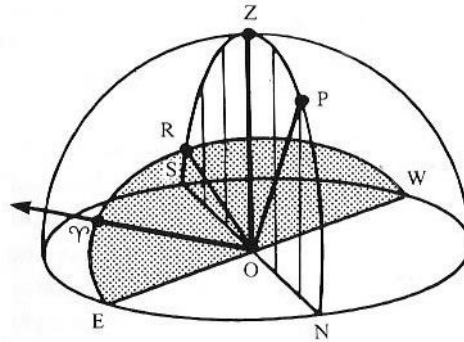


Figure 3: Equatorial Coordinates

Now consider the view seen by the observer standing at O and looking south. Imagine a star, X; its position in relation to the equator could then be calculated in degrees as the distance between the equator at south and the intersection of the great circle line of X, with the horizon. This angle is known as the right ascension, α , and is often quoted in hours by dividing the result in degree by 15 (see figure 4). The second value required to accurately place the star X, is the declination, δ , and represents in degrees how far above or below X lines along XC in relation to the equator.

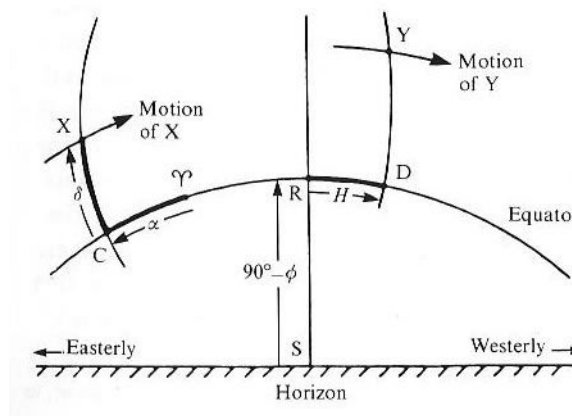


Figure 4: Equatorial Coordinates as seen from the ground in the northern hemisphere

Another useful piece of information that can be extracted from this system is the hour angle (HA). The HA is similar to the right ascension except measures the angle in hours until a celestial body crosses the meridian line RS. For example, in figure 4, a star Y is said to transit (i.e. move onto the meridian line RS) in H hours.

This coordinate system is the primary one used by astronomers to fix objects outside our Solar system, however for object within our Solar system a different system is required which will be covered in the next section.

Ecliptic Coordinates

The Earth's orbit around the sun is called the ecliptic, which forms the basis for this system. Objects within our solar system, e.g. the planets, all follow an orbit around the sun close to this ecliptic plane and therefore it is convenient to use ecliptic coordinates for determining a planet's location.

Imagine the equatorial system from above and lay the ecliptic line on to the equator line. The point at which the two cross is known as the vernal equinox, γ . The angle between the equatorial plane and ecliptic plane is known as the *obliquity of the ecliptic*, ϵ . The two coordinates required to locate a planet, V , are its ecliptic latitude, β and ecliptic longitude, λ . The ecliptic longitude is worked out in a similar fashion to the right ascension in the equatorial system except, the distance along the ecliptic line is measured instead and the answer remains in degrees. The ecliptic latitude is calculated similar to declination from the equatorial system except measuring how high or below the ecliptic the object is (see figure 5).

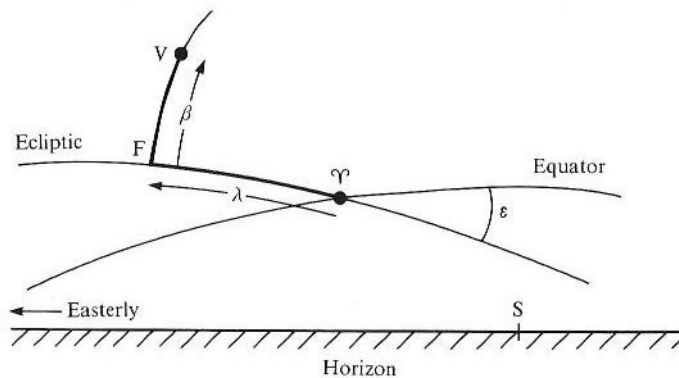


Figure 5: Ecliptic Coordinates as seen from the ground in the northern hemisphere

Coordinate Conversions

It is very common to need the coordinates of a celestial body in different coordinate systems. The most common reference for a celestial body is usually given in the equatorial system, however to the observer on the Earth's surface does not mean much until the coordinates have been converted into the horizon system.

The method for common conversions between the different systems is as follows [B14 p36, p40]:

Equatorial to horizon conversion:

$$\sin a = \sin \delta \sin \Phi + \cos \delta \cos \Phi \cos H$$

$$\cos A = \frac{\sin \delta - \sin \Phi \sin a}{\cos \Phi \cos a}$$

where:

δ = declination

Φ = observer's geographical

H = hour-angle ($H = LST - \alpha$)

a = altitude

A = azimuth

Ecliptic to equatorial conversion:

$$\alpha = \tan^{-1} \left\{ \frac{\sin \lambda \cos \varepsilon - \tan \beta \sin \varepsilon}{\cos \lambda} \right\}$$

$$\delta = \sin^{-1} \{ \sin \beta \cos \varepsilon - \cos \beta \sin \varepsilon \sin \lambda \}$$

where:

α = right ascension

δ = declination

ε = obliquity of the ecliptic (the angle between the plane of the equator and the ecliptic)

β = ecliptic latitude

λ = ecliptic longitude

The obliquity of the ecliptic varies slowly with time and is calculated as follows [B14 p41]:

$$\varepsilon = 23^{\circ}26'21.45'' - 46.815''T - 0.0006''T^2 + 0.00181''T^3$$

where T is the number of Julian centuries since epoch 2000 January 1.5

3.1.4 The Stars

We do not know when humans first started studying the stars, however do know that the science of astronomy was in advanced stages in parts of Europe by 2500 B.C and that the Chinese had astronomical schools by 2000 B.C [B15 p3].

The observations of the sky made during this period, relied entirely on the unaided eye and it was not until technological advances in observation methods, allowed for a more detailed study. During the 16th and 17th century improvements in lens manufacturing and the invention of telescopes led to an explosion of work in the areas of astrometry and astronomy.

Today there are hundreds of star catalogue, of varying size. Each provides many details for each star; the main properties being: the right ascension and declination at a given epoch, the spectral type of the star, the visual magnitude and the apparent motion of the star. These catalogues can be accessed easily from the 'Centre de Donnees astronomiques de Strasbourg' [B10] website.

Visual Appearance

From the properties available, the *visual magnitude* and *spectral type* are of interest when producing images of the stars.

The visual magnitude represents how bright the star, and thus determines how large the star would appear in the sky. Visual magnitude ranges from the brightest stars as having first magnitude and the fainter one having increasing magnitude. The human eye can see stars up to the sixth magnitude without assistance. With the advances in telescopes, stars of magnitude up to 12th and even 13th magnitude can now be found. [B15 p26]

The spectral type of a star indicates its approximate surface temperature. From this it is able to derive what wavelength its electromagnetic emissions are and hence determine its colour [B17]. The classification system is as follows:

Spectral Classification	Temperature (K)	Star Colour
O	30,000-60,000	Blue
B	10,000-30,000	Blue-white
A	7,000-10,000	White
F	6,000-7,500	Yellow-white
G	5,000-6,000	Yellow
K	3,500-5,000	Yellow-orange
M	<3,500	Red

3.1.5 The Sun

The Sun is the most prominent body within our solar system, and the most visible of all celestial bodies from the Earth's surface. It is the nearest star to the Earth, at 91 million miles during its closest approach. All objects within our solar system are controlled by its dominating gravitational pull and as a result all have orbits around the Sun.

This section deals with introducing the fundamental ideas of orbits within our solar system and the positioning the Sun accurately at any given time.

Orbits

The orbits around the Sun can be considered to be elliptical (squashed circles), with two foci, S and S' (see figure 6). As the orbit becomes more squashed, the distance between S and S' increases. If the distance between S and S' is zero, then the orbit becomes perfectly circular.

The amount of squashing of the orbit is measured by eccentricity, e , where a value of 0 indicates no squashing and values approaching 1, a near flattened ellipse.

Imagine a planet V orbiting the Sun located at S (figure 6). As it sweeps around the Sun it is at differing distances from the Sun. The closest position occurs at A , known as perihelion and its most distant at B , aphelion. The size of its ellipse would be defined by its semi-major axis, a , which measures the distance from aphelion and the midpoint of S and S' .

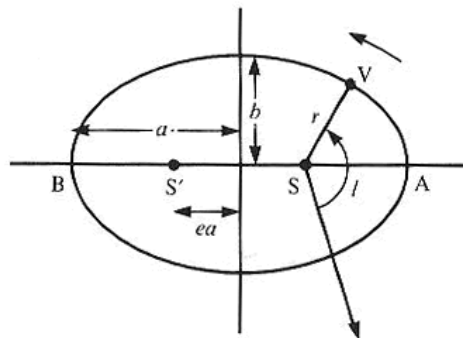


Figure 6: Elliptical Orbits

The Sun's Apparent Orbit

Viewed from Earth, the Sun appears to orbit around us. Therefore we can assume the Earth to be at one of the foci of an elliptical orbit and the Sun to orbit the Earth. At its closest point to Earth the Sun is said to be at apogee and at its furthest point, apogee.

Within our ecliptic coordinate system we would simply have to find the Sun's longitude, λ_{\odot} . Its latitude would remain at zero since its orbit lies along the ecliptic line.

Calculating Sun's Position

Initially we assume that the Sun follows a circular orbit around the Earth. Our starting point is to use the value of mean ecliptic longitude, at a specified epoch. This gives the position of the Sun at that time assuming it was following a circular orbit. From this we add corrections for its true elliptic orbit and for the time at which we would like to calculate its position [B14 p86-87]:

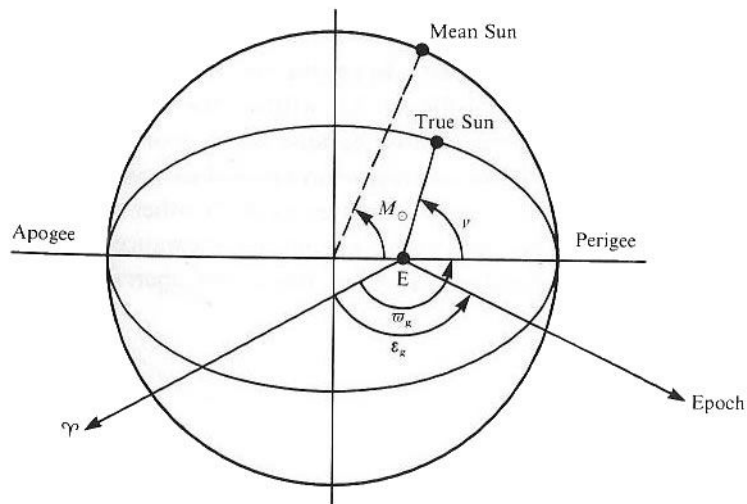


Figure 7: Sun apparent orbit around Earth

Next we calculate the angle, M_{\odot} , this mean Sun has travelled through since our defined epoch:

$$M_{\odot} = \frac{360}{365.242191} D + \varepsilon_g - \varpi_g$$

where:

D = number of days since defined epoch

ε_g = mean longitude of Sun at epoch

ϖ_g = mean longitude of Sun at perigee

We then need to calculate the true motion for the Sun in an elliptic orbit, by finding its true anomaly, v . This is found as follows:

$$v = M_{\odot} + 360 \frac{360}{\pi} e \sin M_{\odot}$$

where:

$e = \text{eccentricity of orbit}$

Finally we can then find the ecliptic longitude, λ_{\odot} as follows:

$$\lambda_{\odot} = v + \varpi_g$$

It is often useful to then convert the ecliptic coordinates into equatorial and then subsequently horizontal to be able to identify where an observer would see the Sun in the sky (see section 3.1.3).

Sun's Angular Size

The Sun is the largest visible object and it is useful to know exactly how large it is at any given time. The angular size allows us to measure the diameter in degrees from the Earth (see figure 8) and is calculated as follows [B14 p92]:

$$\theta = \frac{1 + e \cos v}{1 - e^2} \theta_0$$

where:

$v = \text{Sun's true anomaly as calculated above}$

$\theta_0 = \text{angular diameter of Sun at apogee}$

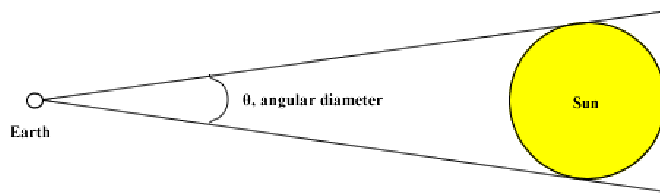


Figure 8: Sun's angular size

3.1.6 The Planets

There are nine planets within our solar system which have been identified so far (ordered by increasing distance from the Sun):

- Mercury
- Venus
- Earth
- Mars
- Jupiter
- Saturn
- Uranus
- Neptune
- Pluto

Each of these planets is bounded by the Sun's gravitational pull and is constrained by an elliptical orbit with the Sun at a focus. Calculating the position of a planet follows similar logic to that of the Sun's apparent orbit around the Earth (section 3.1.5). The major difference is that the plane of orbit of a planet is usually inclined small angle to the plane of the ecliptic.

Imagine a planet, P , within the orbital plane N_1AP with the Sun, S , at the centre (see figure 9) and perihelion A . Now projecting A, N_1, P and N_2 onto an outer sphere with the Sun defined as being the centre, gives a new projection with the planet's orbits being defined at $N'_1A'P'N'_2$. As the P orbits around the sun indicated by the arrow, it crosses N'_2 , the descending node and then N'_1 , the ascending node.

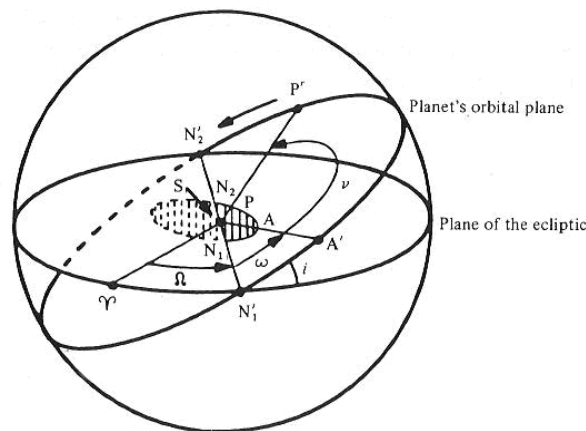


Figure 9: Orbit of planet, P, around Sun, S

This model forms the basis from which the position of any planet can be calculated. Initially we find the position of the planet its own orbital plane. Next project this calculated position onto the plane of the ecliptic giving ecliptic coordinates in reference to the Sun. Finally these coordinates need to be transformed so the reference is that of the Earth.

The details of the method are as follows [B14 p103-106]:

1. calculate D , the number of days since a chosen epoch.
2. calculate mean anomaly, M :

$$M = \frac{360}{365.242191} \times \frac{D}{T_p} + \varepsilon - \varpi \quad \text{deg}$$

where:

T_p = orbital period of the planet in tropical years
 ε = mean longitude of planet at the epoch
 ϖ = longitude of perihelion

3. find true anomaly, v , of planet:

$$v = M + \frac{360}{\pi} e \sin M \quad \text{deg}$$

where:

e = eccentricity of planet's orbit

4. calculate heliocentric longitude, l , and length of radius vector, r :

$$l = v + \varpi$$

$$r = \frac{a(1-e^2)}{1+e \cos v}$$

where:

a = semi-major axis of the planet's orbit.

5. repeat step 2-4 to obtain following values for the Earth:

Earth's heliocentric longitude, L
 Earth's radius vector, R

6. calculate heliocentric latitude of the planet, ψ :

$$\psi = \sin^{-1} \{ \sin(l - \Omega) \sin i \}$$

where:

Ω = longitude of ascending node
 i = inclination of the orbit

7. project values onto ecliptic plane, to find projected heliocentric longitude l' and projected radius vector r' :

$$l' = \tan^{-1} \{ \tan(l - \Omega) \cos i \} + \Omega$$

$$r' = r \cos \psi$$

8. finally transform reference of latitude and longitude to Earth to find geocentric ecliptic latitude, β , and longitude, λ , of planet:

if planet under consideration is Mercury or Venus:

$$\lambda = \tan^{-1} \left\{ \frac{R \sin(l' - L)}{r' - R \cos(l' - L)} \right\} + l'$$

else

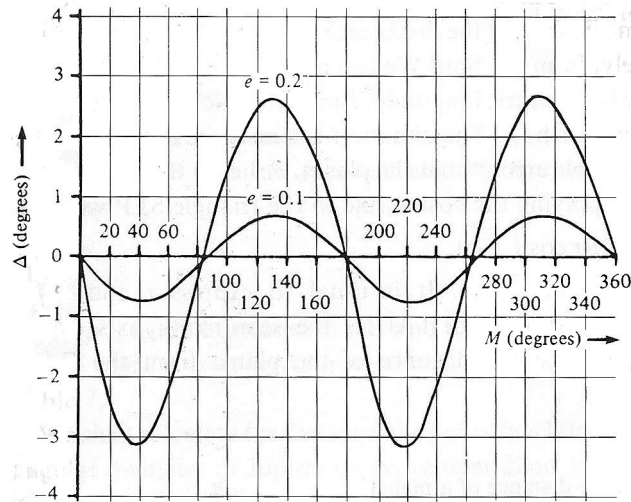
$$\lambda = 180 + L + \tan^{-1} \left\{ \frac{r' \sin(L - l')}{R - r' \cos(L - l')} \right\}$$

$$\beta = \tan^{-1} \left\{ \frac{r' \tan \psi \sin(\lambda - l')}{R - r' \cos(L - l')} \right\}$$

Limitations of method:

The method outlined above does provide the planet position to sufficient accuracy, with right ascension and declination being within a few minutes of the true position for near circular orbits. The errors created are due to approximations made in the elliptical motion by finding the anomaly, v , as

$v = M + \frac{360}{\pi} e \sin M$. The true anomaly should be $v - \Delta$, where Δ is shown to vary as follows:



where:

e = eccentricity of planet's orbit

It can be easily seen that as the orbit becomes more flattened (larger eccentricity), the resulting errors can become quite large and not suitable for positioning planets to reasonable accuracy. Mercury and Pluto have eccentricities greater than 0.2 and are therefore very susceptible to errors using the method outlined above.

3.1.7 The Moon

With the moon lying closest to the Earth from all the celestial bodies it is by far the most visible object in the night sky. Its movement against the night sky is so rapid that its displacement against the background stars can easily be seen.

The Moon's position is the most difficult to predict accurately due to the gravitational effects of both the Sun and Earth. It orbits around the Earth and follows the Earth's orbit around the Sun, as shown in figure 10 below.

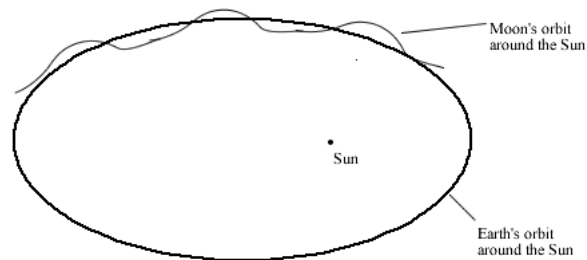


Figure 10: Moon's orbit

This results in a long complicated method in order to accurately find its position, based loosely on the methods of calculating a planet's position. A method for finding its position is outline below from [B14 p142-145], for details regarding in depth reasoning consult [B14 p138-143]:

1. calculate the Sun's ecliptic longitude, λ_{\odot} , and mean anomaly, M_{\odot} , as shown in section 3.1.5
2. calculate the Moon's mean longitude, l , given by:

$$l = 13.1763966 D + l_0$$

where:

D = number of days since epoch to required date

l_0 = Moon's mean longitude at the epoch

3. calculate the Moon's mean anomaly, M_m , given by:

$$M_m = l - 0.1114041 D - P_0$$

Where:

P_0 = Mean longitude of the perigee at the epoch

4. the ascending node's mean longitude, N , given by:

$$N = N_0 - 0.0529539 D$$

Where:

N_0 = Mean longitude of the node at the epoch

5. calculate the corrections for evection, E_v , the annual equation, A_e , and a third correction, A_3 .

$$E_v = 1.2739 \sin(2C - M_m)$$

$$A_e = 0.1858 \sin(\lambda_{\odot})$$

$$A_3 = 0.37 \sin(\lambda_{\odot})$$

Where:

$$C = l - \lambda_{\odot}$$

6. from corrections, find the Moon's corrected anomaly, M'_m :

$$M'_m = M_m + E_v - A_e - A_3$$

7. find the correction for the equation of the centre:

$$E_c = 6.2886 \sin(M'_m)$$

$$A_4 = 0.214 \sin(2M'_m)$$

8. find the value of the Moon's corrected longitude, l' :

$$l' = l + E_v + E_c - A_e + A_4$$

9. apply final correction for variation, V , to Moon's longitude:

$$V = 0.6583 \sin 2(l' - \lambda_\odot)$$

$$l'' = l' + V$$

10. finally calculate the ecliptic latitude, β_m , and the ecliptic longitude, λ_m as follows:

$$\lambda_m = \tan^{-1} \left\{ \frac{\sin(l'' - N') \cos i}{\cos(l'' - N')} \right\} + N'$$

$$\beta_m = \sin^{-1} \{ \sin(l'' - N') \sin i \}$$

where:

$$N' = N - 0.16 \sin(M_\odot)$$

Accuracy

The method outlined above gives the ecliptic coordinates with up to 1/5th degree of error, which may cause problems when trying to test solar eclipses accurately on certain dates.

3.1.8 Projections

The coordinate systems outlined in section 3.1.3 represents positions of celestial bodies in a three dimensional universe, however for output onto a screen or paper, these coordinates need to be transformed into a new system. This kind of transformation is known as a map projection.

There is huge number of map projections available today, each one used for a different reason [B18]. The Mercator projection is one of the earliest used projections in producing maps of the Earth. Astronomers however are interested in maps of the sky, with the Polar and Zenith projections being popular choices.

Mercator Projection

Gerardus Mercator devised this projection during the mid 16th century, during his attempts to produce navigable maps of the world. It is known as a cylindrical projection, in that the meridians are spaced equally. It is easiest to imagine the map being generated by a cylinder being placed around the Earth, with the length of the cylinder lying parallel to the Earth's rotational axis. Lines perpendicular to the cylinder intersecting the Earth's surface then generate the points of the map [B19].

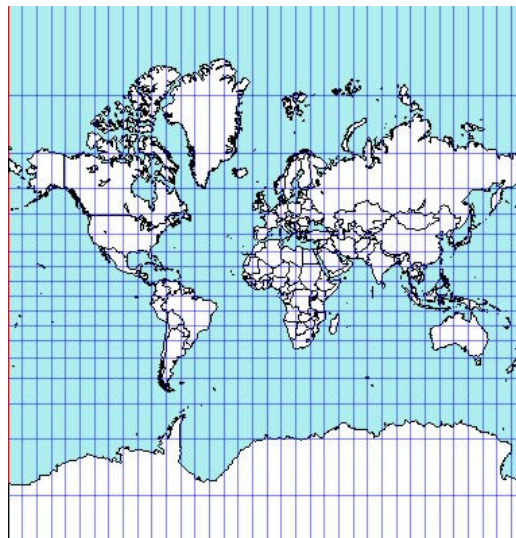


Figure 11: Mercator project of Earth [B19]

Polar Projection

This projection results in a map showing either the entire northern or southern celestial hemisphere. The centre of map is always either the North or South Pole, with the edge of the map representing the Earth's equator. For the purpose of this project such a projection is not of primary use, since the view needs to simulate what an observer would see looking into the sky from a specific location on Earth.

Zenith Projection

This projection is ideal for a planetarium, as it is similar to the horizontal projection in its approach. Imagine the horizon coordinate setup from section 3.1.3, now take all the bodies on the celestial sphere and map them

perpendicularly onto the horizon plane giving the projection results. Figure 12 shows how a star, S, maps onto the projection based on its horizontal coordinates.

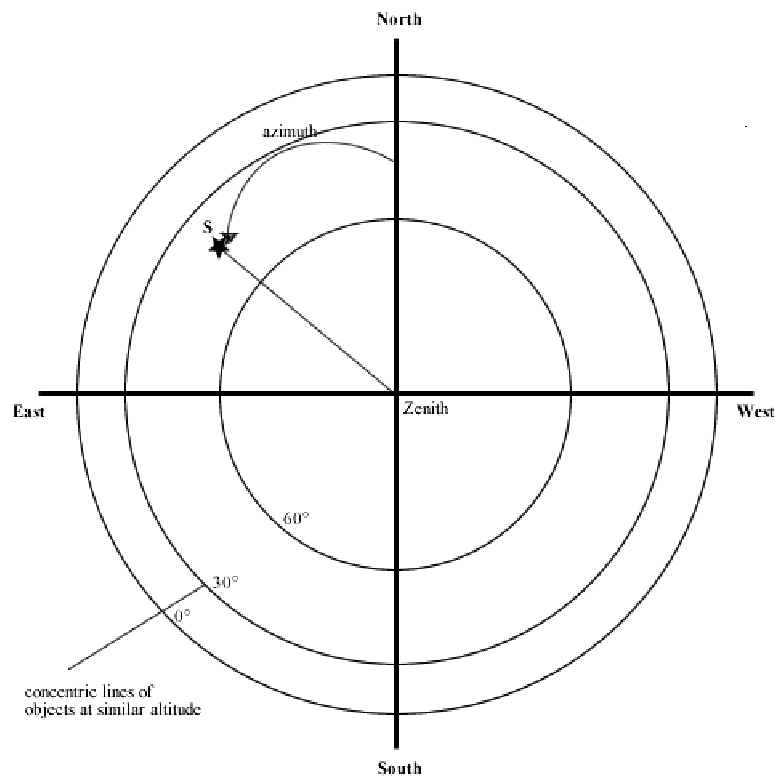


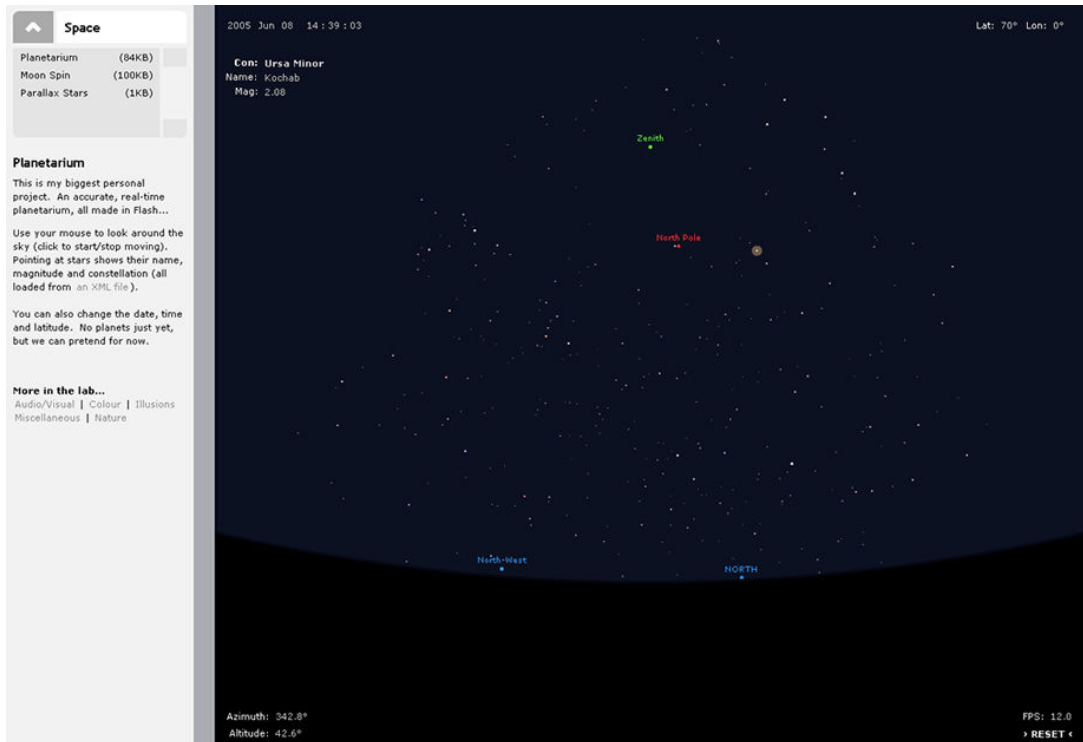
Figure 12: Zenith projection

3.2 Existing Solutions

There are many existing software solution simulating a planetarium, however there are few online solutions. This section outlines a few and provides a brief analysis of each.

Neave's Planetarium

www.neave.com/lab/space/

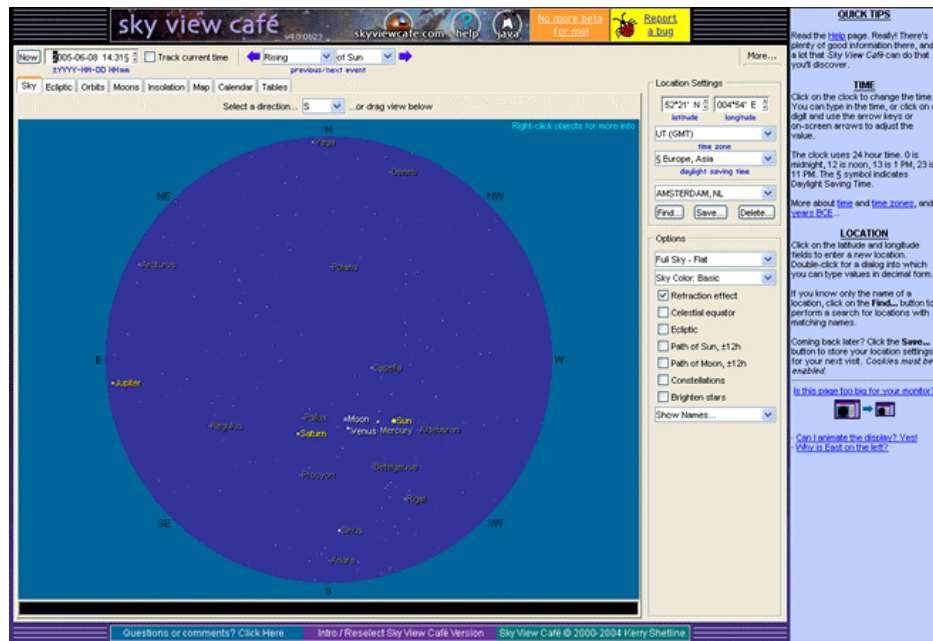


Neave's Planetarium is based entirely on Macromedia's Flash plug-in. It has a very clean interface and it is simple to use. The images generated on the screen are vector based and as a result scale nicely for all screen sizes without distortion. The plug-in allows for a high-level of interactivity as the user can adjust the view without any delay of a server response.

The system does however feel incomplete, as there is no indication of planet positions or other bodies in the solar system. The stars are all white and have no information regarding spectral type or colour.

Sky View Café

www.skyviewcafe.com



This is one of the most complete web-based planetariums encountered on the web. The application is run from within a java applet window and is packed with interesting features.

The interface appears to be complicated in comparison to Neave's, however allows for much greater customisation in the images generated. The use of checkboxes down the right side, allow much configuration of the visible objects on screen. The location settings allow a user to specify a location based on the name of a city and not rely solely on latitude and longitudes inputs.

Other features of this application allows users to:

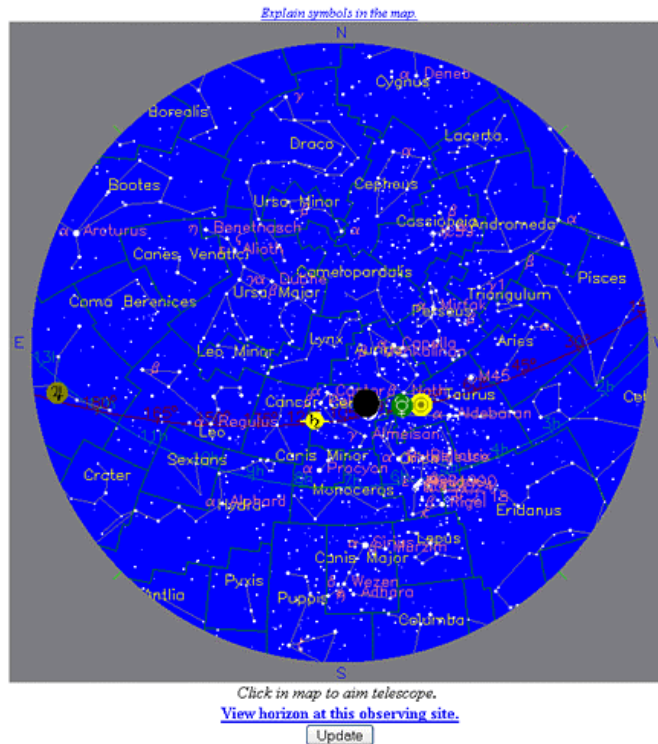
- View the orbital paths of the planets in more detail.
- Calculate rise and set times for the Sun, Moon, and planets.
- Check positions of Jupiter and Saturn's moons.
- Provide phases of Earth's moon.
- Produce a perpetual calendar with astronomical events.
- Produce ephemeris tables for the planets within our solar system.

The use of an applet has allowed for a high level of interactivity with near no delay between user requests and display updates, showing a very efficient implementation.

YourSky

www.fourmilab.ch/yoursky/

Sky above 47°N 7°E at Wed 2005 Jun 8 13:36 UTC



YourSky is different from the previous examples since the images generated are not vector based. The images are generated by the server, based on the user's input and then sent back to the user in GIF format. This keeps the bandwidth requirements of the application at a minimum, with the image typically 30 kilobytes and the associated html code at a few kilobytes.

There is a comprehensive set of options available customising the image generated, however the presentation is very simple and gives the impression of an incomplete product.

This system has advantages over the previous example by allowing the user to choose from 3 different viewpoints:

- 'Sky Map' – draws the entire sky on a stereographic projection based on a location and time
- 'Horizon View' – provides a view of sky shown in relation in the horizon line.
- 'Virtual Telescope' – allows users to move view around sky at different magnifications, specifying the aiming point of a virtual telescope.

Overall this is also a powerful system, with few requirements from the browser (no plug-ins needed), and connection bandwidth.

3.2.1 Existing Solutions Summary

These solutions outlined all offer useful features, however there is not an existing planetarium on the web available that combines all the features and maintains a high level of interaction. Sky View Café may appear to be a

complete solution, but lacks the ability to pan, which is present in Neave's and YourSky. It also lacks a true zoom function, only providing a zoomed image from the NASA database. The colours of the stars are also never visually represented in these examples, despite the spectral types being available.

4. Decisions

This chapter aims to outline the decisions made early during the development, and the reasons for choices made.

4.1 Technology

The final product of this project aims to target web users and as a result a web technology needs to be used. Over the years a wide range of web based technologies have become available and a choice had to be made regarding the best suited for this project.

4.1.1 Technology used

The chosen technology played an important role in the final product and trade-offs had to be made regarding two main aspects:

- i. Level of interaction – how easily can the user make input changes and see the resulting changes occur
- ii. Performance – could the technology cope with a vast amount of data and represent this on a screen without delay

PHP – Hypertext Pre-processor [T1]

The choice was made to use the server scripting language, PHP, as main the programming language to create the astronomy models detailed in the background. This appeared to be an ideal choice with its strong mathematical capability (covering all the desired trigonometry functions) and providing a good support for image manipulation via its GD module.

Portable Network Graphic (PNG) [T2]

PNG was chosen as the output format for the image generated by the PHP GD module. It is a lose-less compression format, which maintains the image quality. This was ideal, as experimentation with other lossy compression formats, showed that even a slight decrease in image quality led to a difficulty in seeing small, one pixel stars accurately due to blending with nearby pixels. The lossless bitmap image (BMP) was also considered, however even a reasonable sized image of 500 pixel by 500 pixels @ 24bit colour, resulted in a massive file size of 732kb.

PostgreSQL [T3]

There can be huge amounts of data associated with a star catalogue and the only suitable option was to store this within a relational database. PostgreSQL was suited with its strong support in communication with PHP scripts.

HTML & Cascading Style Sheets (CSS)

HTML is the only option suited to working with PHP, and can pass user requests to the PHP script via 'POST' and 'GET' methods via the HTTP protocol. The use of CSS allows customisation of the visual aspects of the interface.

4.1.2 Other Possible Technologies

There were three alternative options being considered for the implementation language for the main modelling module:

1. Java Applet
2. Tomcat & Java Servlet
3. Macromedia Flash & Actionscript

However the reasons for against using them are as follows:

Java Applet

This may have allowed for more interactivity via the use of built-in GUI widgets, however the decision was made against using applets due to increased bandwidth demands placed since the applet would have to talk directly to the database and thus pass large amounts of data across the internet, depending on the star catalogue size.

Tomcat & Java Servlet

This could have been an ideal choice, offering similar benefits to that of PHP, however the unreliability of the Tomcat made it second to PHP.

Macromedia Flash & Actionscript

Flash would have allowed for a sophisticated interface to be built with its strong emphasis on designing from a visual point of viewpoint. However its performance using Actionscript can be rather slow compared to java or PHP when dealing with large data structures (star catalogues).

4.1.3 Technology Summary

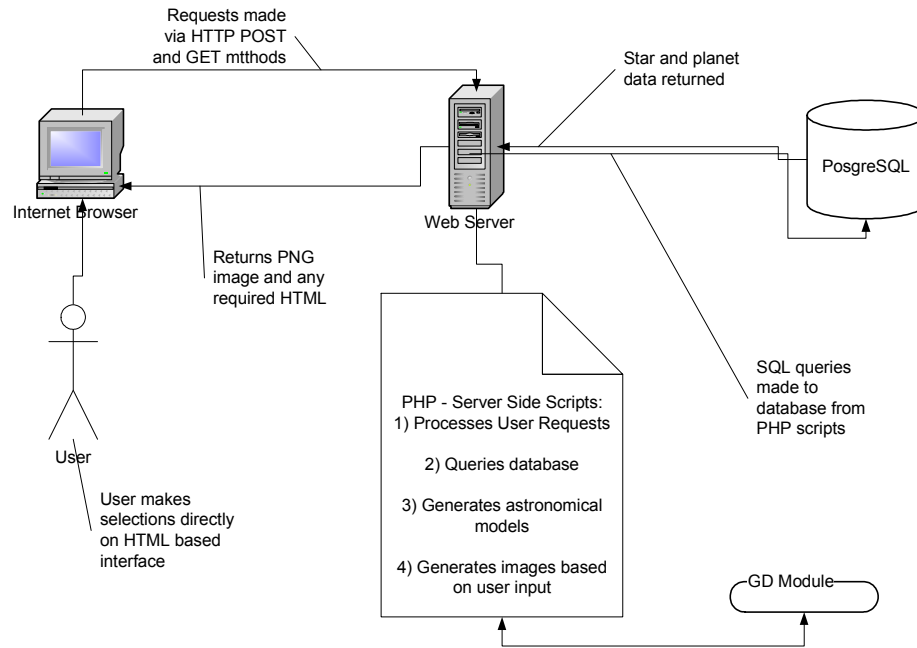


Figure 13: Technology overview

Figure 13 above, shows how these chosen technologies are linked together to provide a solution that has the required performances level with the appropriate level of interaction capabilities.

4.2 System Objectives

A virtual planetarium needs to meet many requirements for it to function in a user friendly and satisfactory way. The main one including providing reasonable accurate images of the sky based on a location and time. The expectations of this system and features hoped to be included in the final implementation are discussed in this section.

4.2.1 System Specification

The following set of functionality was set out from the start:

- Collect input from user regarding two main variables:
 - The observer's location on the Earth
 - The observer's local time and date
- Provide the user with an image representing the sky at the specified location and time including the following data:
 - Star name/identifier
 - Sun/Moon location
 - Constellation grid labels and lines
 - Celestial grid for reference
- Allow users to change the view presented by the following methods:
 - Re-centring of the view based on any location specified by a clicking action on the image.
 - Panning of the view by moving the centre horizontally or vertically based on user's decision.
 - Zooming in/out within the image on the current centre point

- Present the user with information regarding a star of their choice by a clicking action on the desired star.
- Stars represented within the image, based on their magnitude and spectral type to determine their size and colour respectively.
- Planets displayed at an accurate size and position

4.2.3 Star Catalogues

Many star catalogues are available today, each containing varying amounts of data for every star listed. The Centre de Donnees astronomiques de Strasbourg [B10] provides the most extensive range of catalogues to pick from. Each catalogue can be downloaded as a *.DAT* file and a provided *readme* file explains the byte-by-byte structure of the file. This allowed for any suitable catalogue to be used as a data source for displaying the stars.

Initially, the widely used Bright Star Catalogue (BSC) was parsed and entered into the postgresQL database. BSC provided all the necessary entries to satisfy the specification, and with a size of 9110 stars, provided a good starting point to scale the system from later in development.

It was quite apparent further into development that zooming several magnifications within a section of the sky, did not provide a sufficient star density using BSC (see figure 13), and an alternative had to be found.

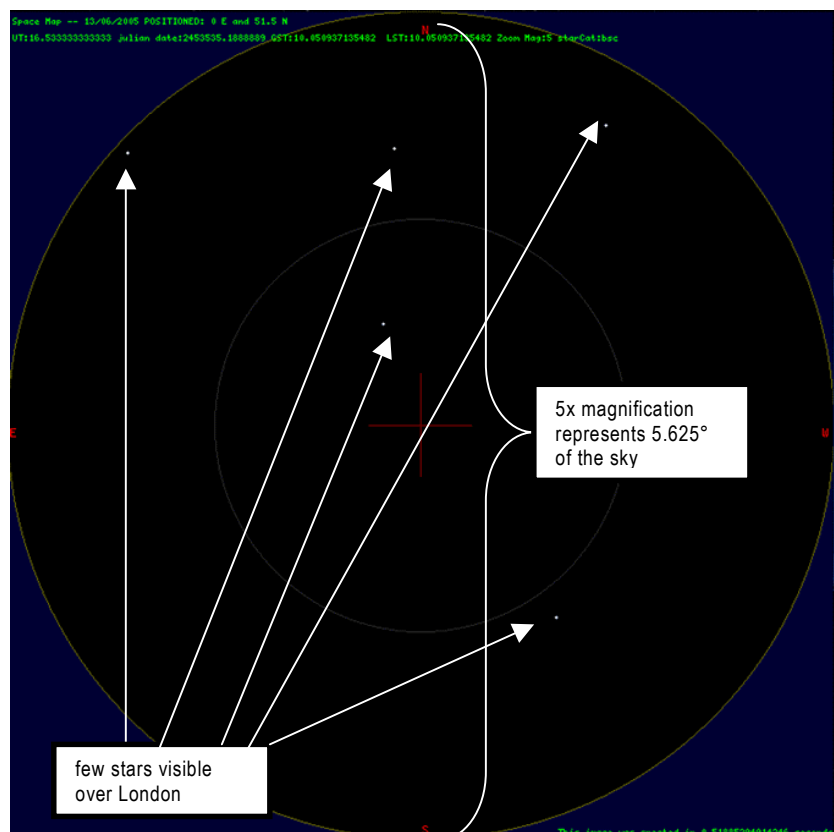


Figure 14: Bright Star Catalogue @ 5x magnification

The 'Hipparcos' [B21] and 'All-sky' [B20] catalogues, were two chosen

alternatives that met the requirements outlined above. Hipparcos is a medium sized catalogue, compiled between November 1989 and March 1993, with the aid of the Hipparcos satellite and contains 118,218 stars. The substantial increase in data, immediately made a difference (see figure 15), with a visible increase in star density

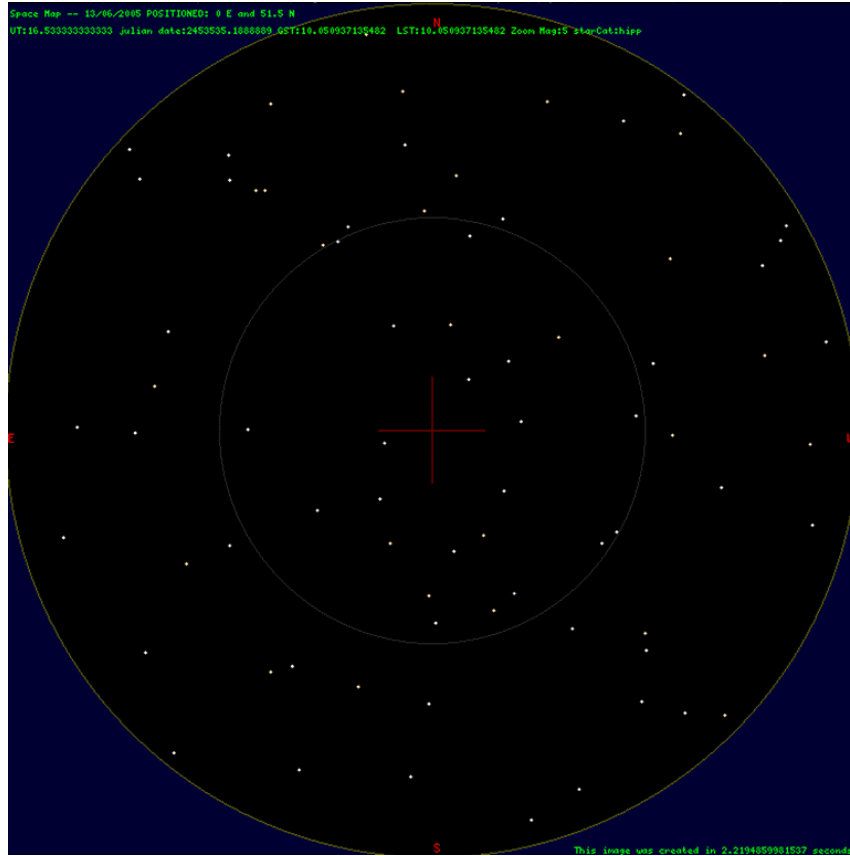


Figure 15: Hipparcos Catalogue @ 5x magnification, at same location and time as figure 14

The All-sky catalogue is one the largest catalogues to date, with 2,501,304 stars included up to magnitude 14. It is composed from the combination of several recent high precision catalogues; Hipparcos, Tycho-1, Tycho-2, ACT-RC and TRC to name a few. Figure 16 shows the stars density for All-sky for comparison with figures 14, again showing a dramatic improvement.

The large set allowed for a high star density to be seen at even the very high magnification levels, whereas the previous two catalogues quickly showed empty space beyond a magnification level of 7.

Implications

The use of All-sky created complications due to its sheer size⁴ and querying of a such a large dataset caused bottlenecks and lead to extremely poor performance or complete failure of the PHP scripts due to its 30 second time out expiry on the web server. This problem and its solution will be discussed within the implementation section of this report.

⁴ All-sky .dat file totals 415mb compared to Hipparcos's 14.4mb and BSC's 2.2mb

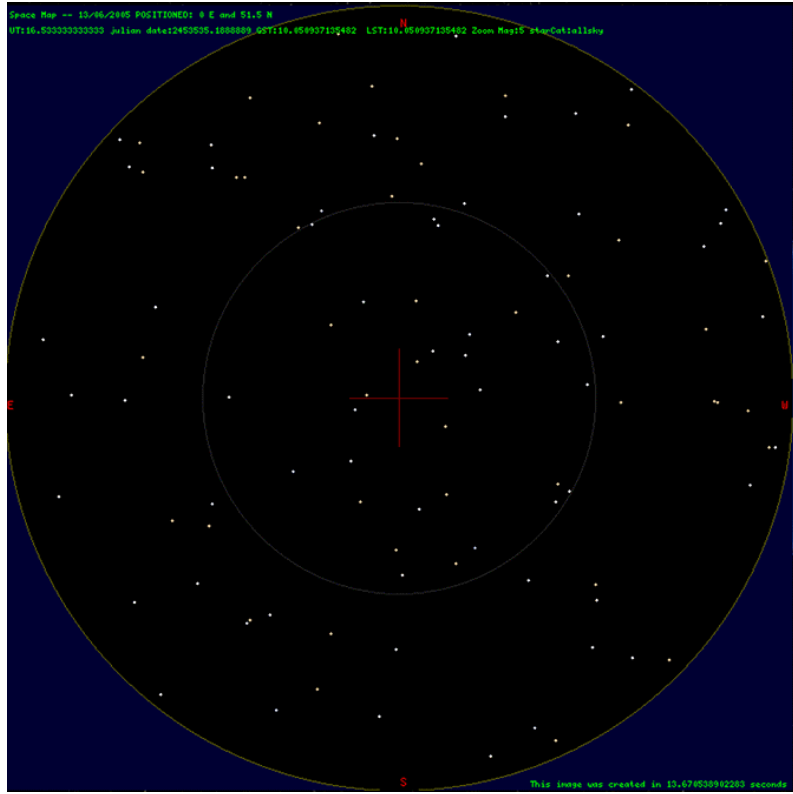


Figure 16: All-Sky Catalogue @ 5x magnification, at same location and time as figure 14

5. Design

This chapter's aim is to provide insight into the design stages of the final application. It highlights how an initial solution was devised from the requirements set out earlier.

5.1 System Overview

The architecture of the system is based around that of a typical database driven web application and as a result lead to the creation of three distinct components:

Database structure – Creation of tables and associated attributes, to hold star catalogues and orbital data for the nine planets.

Server-side scripts – responsible for processing user's input, in order to generate the appropriate image

Client-side front-end – Interface for a user to make image requests and view manipulations

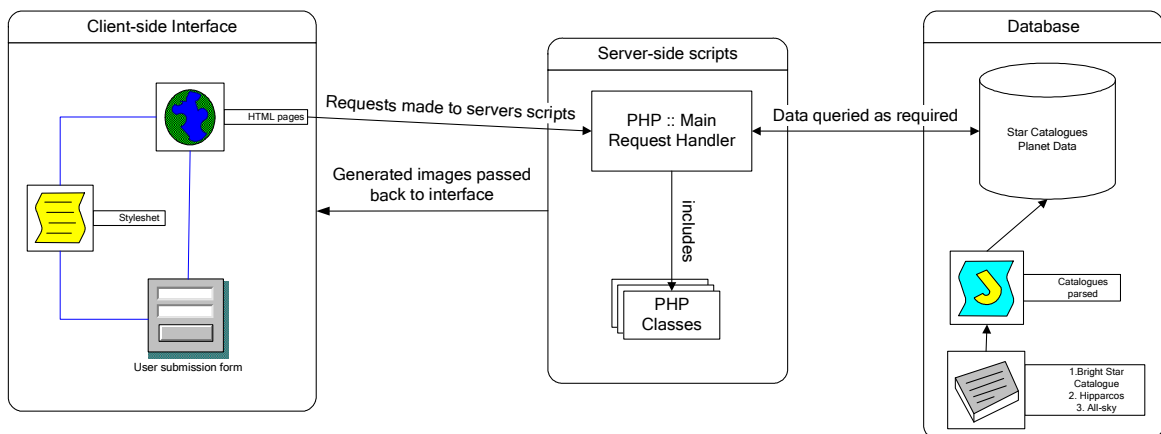


Figure 17: Overview of the three major components

Each component's design and implementation will now be considered through the next few sections, highlighting how the requirements directed the design and eventual implementation

5.2 Database

This is the smallest of the three components and its sole purpose is to supply the server-side scripts with required positional data of stars, constellations, and planetary orbits.

5.2.1 Tables & Structure

The schema was very simple (see figure 18) since no relationships between the tables existed. As a result it was considered to store the data in a flat file

structure, i.e. leave them in their original .dat file structure. However the larger catalogues would have suffered a huge performance hit processing the entire file, on every request, so to increase performance, optimisation were made using highly specific queries which could be performed on the database (see later for optimisations).

To ensure the fastest possible performance, indexes were created on the star catalogue's right ascension and declination. This lead to faster performance when querying the tables for stars at specific equatorial coordinates.

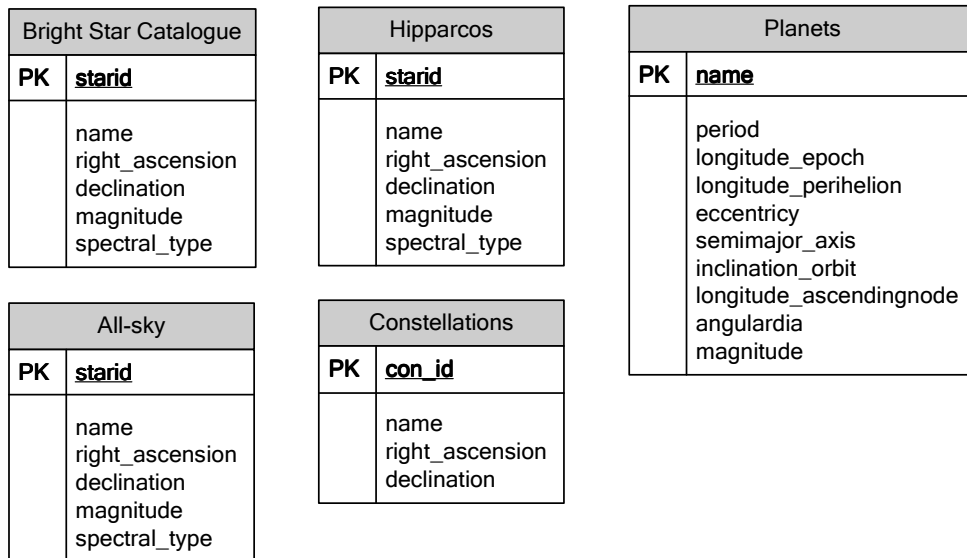
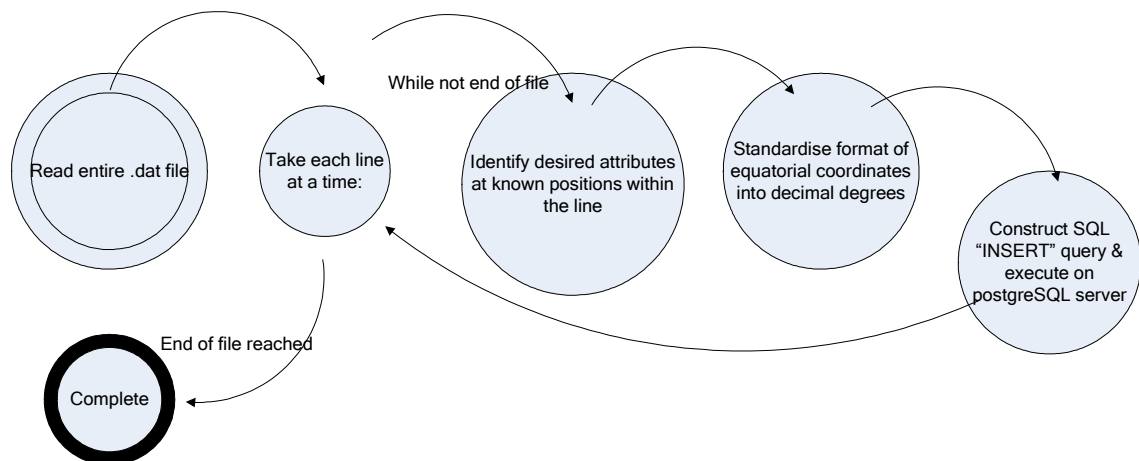


Figure 18: Database schema

5.2.2 Catalogue Parsing

A simple java class was created to read in the .dat files and standardise any differences in units or string layouts between the different catalogues. The data was then inserted directly into the appropriate tables.



The current implementation can support almost any star catalogue with the only requirements being that every entry has an equatorial position at epoch J2000 and a visual magnitude. By default a star is shown as white if no spectral type is available and an automatically generated integer is assigned for stars lacking a name.

5.3 Client-side front-end

The interface presented to the user needed careful consideration in its design, as that is the only visible aspect of the project. There were three main sections of the interface to deal with; the HTML form responsible for collecting the user input regarding time and position of the observer, the facility to allow changes to be made to the view, and finally an area to display the output.

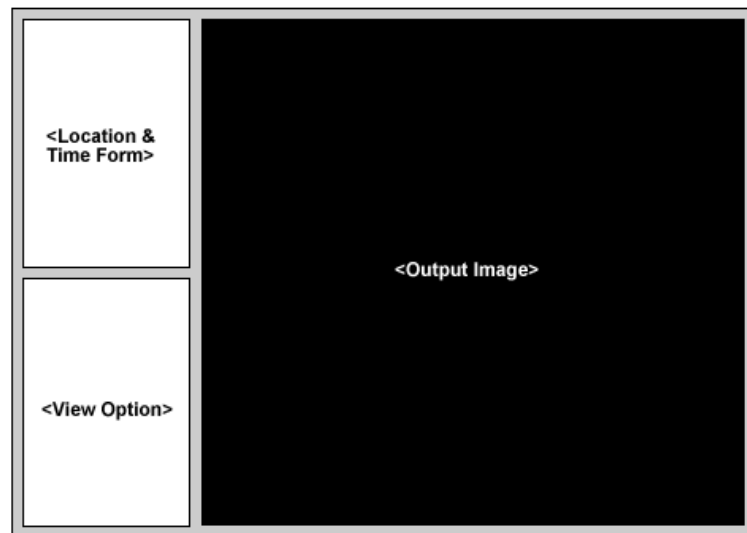


Figure 19: Interface layout

Figure 19 shows the initial design layout. The primary goal was to minimise user scrolling, to ensure a smoother operation without the distraction of trying to find items on a long page. The output image is always a perfect square due to the zenith projection producing circles, which leaves space down the left hand side to occupy with the user input form and view options.

A CSS file controls the visual appearance of the interface, with the form elements in HTML. This places certain requirements on the user Internet browser; specifically the support of CSS classes, however a lack of support will not render the interface useless, only visually less vibrant. Most browsers today have strong support for CCS; the main ones used for testing include:

- Internet Explorer 5.x/6.x
- Mozilla 1.7.x
- Mozilla's Firefox 1.x
- Safari

No additions plug-ins are required by the browser for the interface to work, such as the presence of a java virtual machine, and should run on any browser with support for displaying the PNG image format.

5.3.1 Interface Generation

The user's Internet browser is sent dynamic information, based on their

previous inputs and selections, by the PHP server-side scripts. HTML created by these scripts, depends strongly on what situation the user is in.

When the user first visits the site no image is displayed until the basic required information of location and time is filled out. To aid this process default values are filled in setting the location to London and time and date to the current values. From this setup the user can then proceed with the defaults or change these values and then submit the values. The server-scripts validate the input and then present the new view to the user.

Once an initial view is created, the user can then either:

1. Generate a new view by resubmitting a new location and time
2. Alter the view presented by:
 - Zooming in/out on the centre of the image
 - Panning the view along either the horizontal or vertical axis.
 - Re-centring the view by left clicking on any location on the image.
 - Altering the star catalogue used to draw the stars
 - Enabling or disabling viewing of constellations, celestial grid and star names.
3. Obtain star information by changing the action of the left clicking action and then selecting any visible star.

As a user makes changes the state is stored between submissions so the forms on the left hand side always indicate what location and time the image is based on.

5.3.2 Zooming and panning controls

The zooming and panning control's (see figure 20) main aim is to help position the virtual telescope in the sky. The panning controls are placed around their respective locations around the image based on their axis of movement. The zoom buttons then control the magnification level of this virtual telescope.

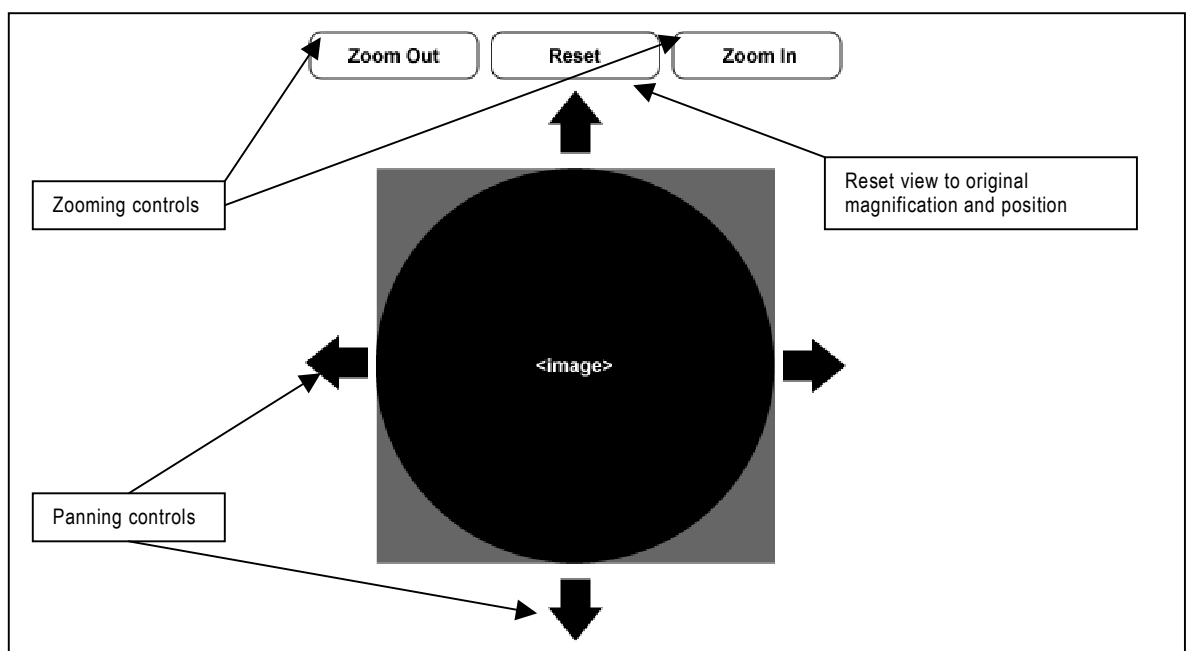


Figure 20: Panning and zooming interface design

5.4 Server-side scripts

The server-side scripts have been written entirely in PHP, using the GD module where necessary in order to help create images. PHP provides support for an object oriented approach despite it being a scripting language and thus allowed for a structured approach in server-side design.

Figure 21 shows how the PHP scripts were used to systematic, collect and validate input, make appropriate queries on the database, generate the models required, and finally output an image and the updated interface.

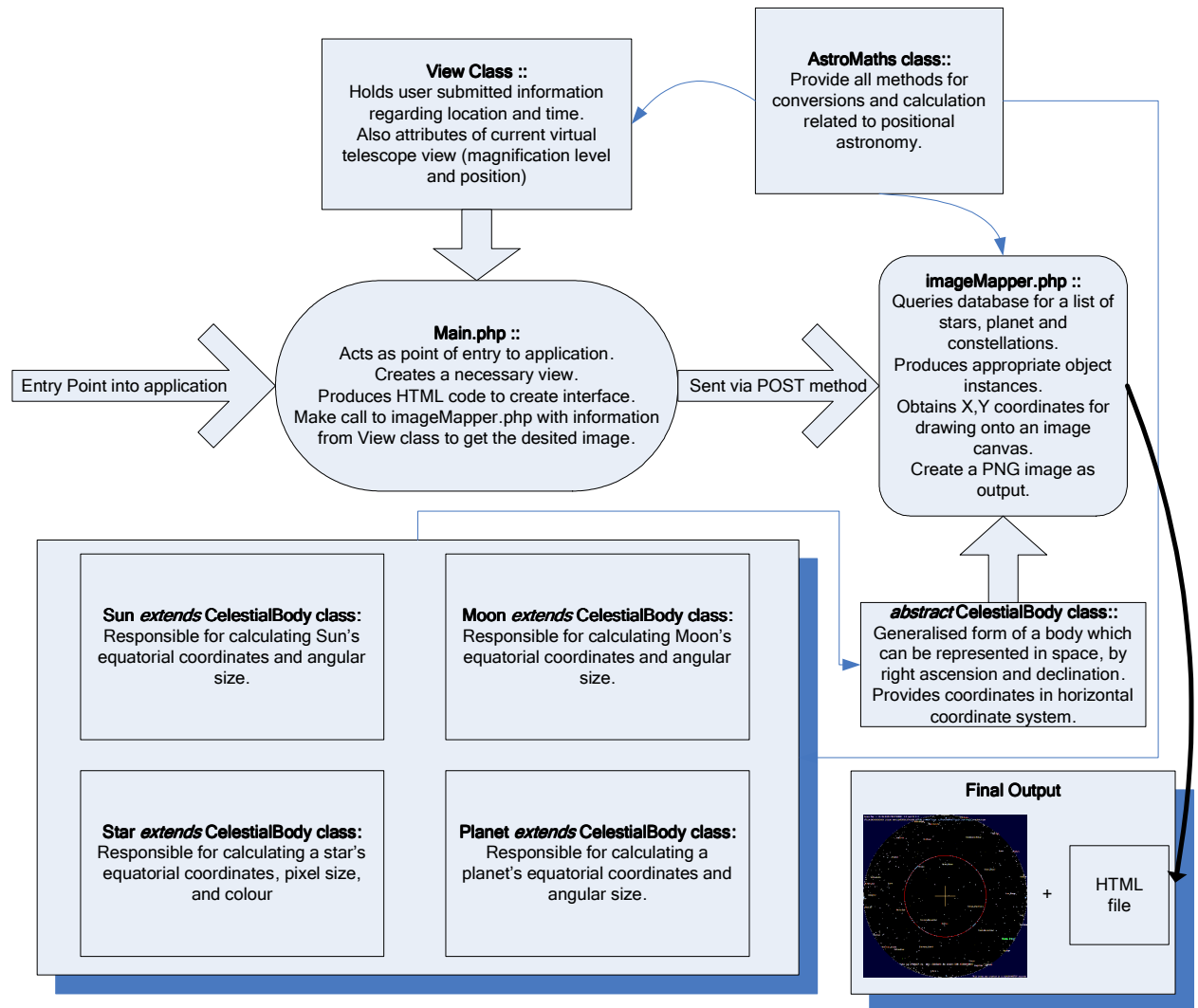


Figure 21: Server-side Overview

This approach allows the system to be highly extendible, allowing scope for infinite celestial bodies to be added to the final image simply by extending the *CelestialBody* class with a new desired class.

The algorithms discussed in the background have been implemented within the *AstroMaths* class with the more specific algorithms for calculating position of Sun, Moon and planet; performed in its rightful class. By implementing the common procedures within *Astromaths*, allowed much reuse of its methods throughout almost all the other classes and significantly reduced development time.

6. Implementation

This chapter highlights important sections of the implementation and where most of the time was spent in during the development. Interesting issues that arose are also mentioned with the solution used to overcome them.

6.1 Maintaining State

Web based projects are always subject to the same issue of being stateless, i.e. they maintain no information between browser requests. Many way are available to overcome this stateless situation:

- **Cookies:** Small pieces of information stored at the client-side within a temporary file that can be read by the web server during every user request.
- **Hidden variables:** HTML input forms are embedded with hidden variables, identifying a user or holding values, which are sent to the server script in every request.
- **PHP Sessions:** The web server stores a file on the server which is uniquely identified by the by a session id, which is generated on the first request made to a PHP script.

Sessions was the best choice to use since an entire class object could be stored between user requests, holding important information regarding, user's current location and time, virtual telescope position and magnification level. Sessions have an advantage over cookies since cookies are not supported by every browser and on those supported, users can disable their use due to privacy issues. Sessions on the other hand are implemented on the server side allowing for better controls and prevent blocking at a user level. The use of hidden variables was dismissed due to unnecessary HTML being sent to the user.

In the final implementation the instance of the View class (see section 5.4), was stored between user requests. This allowed the interface to keep track of previously entered fields and fill them in on further requests, providing a smoother experience without the worry of remembering previously entered data for the user.

6.2 GD Graphics Library

GD is an open-source graphics library, designed for programmers allowing them to dynamically create images. Its most popular use in PHP is in the creation of dynamic charts for web sites from live data [T5].

6.2.1 Basic Drawing Functions

The library calls are made from within PHP scripts. The following code segment shows an example how to draw basic shapes:

```

        <?php
// creates a blank image instance with dimensions 400x300 pixels.
$image = imagecreate(400, 300);

// choose a colour for the circle, defined its red, green, blue
values.
$col_ellipse = imagecolorallocate($image, 255, 255, 255);

// draw the white circle, at location (200,150) with width and
height of 200 pixels, using the colour $col_ellipse
imagefilledellipse($image, 200, 150, 200, 200, $col_ellipse);

// write a string at the top left
imagestring($im, 5, 10, 10, "circle", $col_ellipse);

// draw a white line
imageline( $image, 50, 50, 50, 100, $col_ellipse )

// output the picture as type PNG
header("Content-type: image/png");
imagepng($image);

php?>

```

The resulting image of this code segment is shown in figure 22 below.

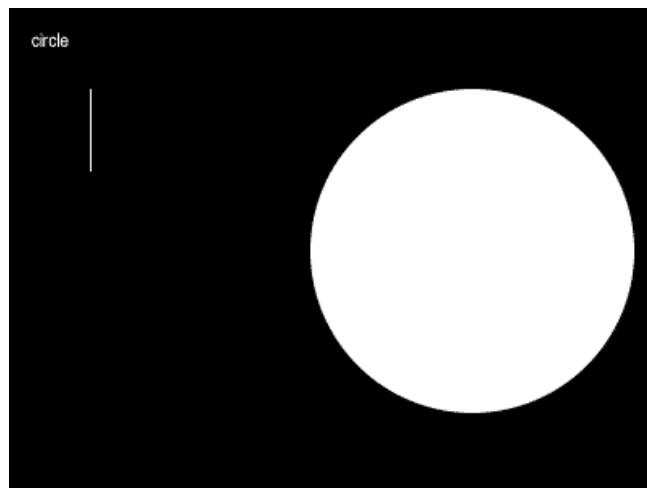


Figure 22: Results of basic GD functions

6.2.2 Final Graphics Engine

The final implementation relied on the creation of a *CanvasTools* class, which had the primary function of doing the majority of the drawing work. Some of the methods involved overlaying the compass headings, generating the correct sizes of star depending on their magnitude, and merging images of planet surfaces after rescaling, based on calculated angular diameters onto the final image canvas.

These methods were composed of the basic functions outlined above in section 6.2.1 and allowed the creation of a sky map as seen in the final product.

6.3 Algorithm Issues

The majority of the programming time was spent converting the many formulas and algorithms detailed in the background, into PHP code. Even though the methods presented, appeared to be a list of formulas, their implementation into a programming language caused a few problems.

6.3.1 Inverse tan functions

Some formula rely on calculating the inverse of the tan (\tan^{-1}), and this can introduce some wrong values, using the straight implementation in PHP of `atan($X)`. This is due the fact that there is ambiguity present as to which quadrant of the trigonometric circle the answer should lie in.

The inverse tan is usually performed on a fraction within the methods, and it was discovered that by using the different PHP function `atan2 ($x, $y)` instead, the errors could be eliminated. This different function performs the inverse on the fraction ($\$x/\y), however takes into consideration the signs of both $\$x$ and $\$y$, performing correction and giving a result within the correct quadrant.

6.3.2 Testing

The nature of the astronomy models being simulated relied heavily on the accuracy of every component. Small errors generated by even simple methods (e.g. the calculation of the Julian date), had to be identified early and rectified. This was best achieved by testing every method with worked examples from the core textbook [B14] used and from online calculators, as implementation progressed.

This ensured minimised backtracking through code in order to identify bugs from earlier work.

6.4 Zooming and panning

The zooming and panning features posed an extra level of complication to the project. The easiest solution would have been to take the original images created and simply enlarge and crop then to the desired view. However this would lead to incorrect images, because the image being generated is already that of a zenith projection that is very susceptible to distortion around the edges.

6.4.1 Zooming Solution

Zooming was achieved by generating the original image as before, however the positions of the object being drawn are scaled by the zoom factor in each direction from the centre of the image (see figure 23).

The Sun, Moon and planets are subsequently drawn larger, with their original pixel size increased by the zoom factor. However for the stars it is not as simple as increasing their size by the zoom factor since zooming does not necessary double the size of a star. Instead a method was written which provides a reasonable size for a specific star depending on it magnitude and the magnification of the view.

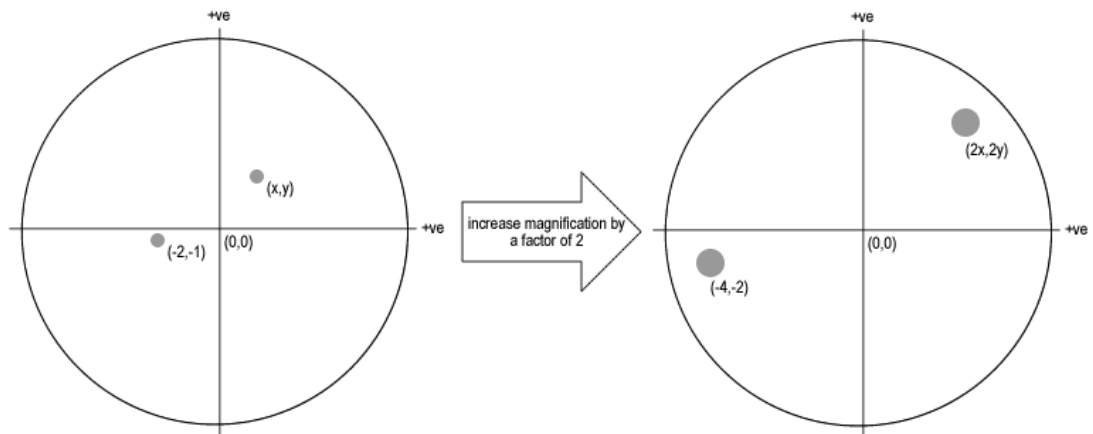


Figure 23: Zooming Solution

6.4.2 Panning Solution

Shifting the original view presented to the users was achieved by the creation of a virtual telescope. This virtual telescope maintains a pair of values indicating where on the Earth the apparent image is being projected. The zenith projection always keeps the observer zenith at the centre of the image so in order to pan the view to one side, the observer virtual position must shift in accordance to keep it directly beneath the new zenith (see figure 24).

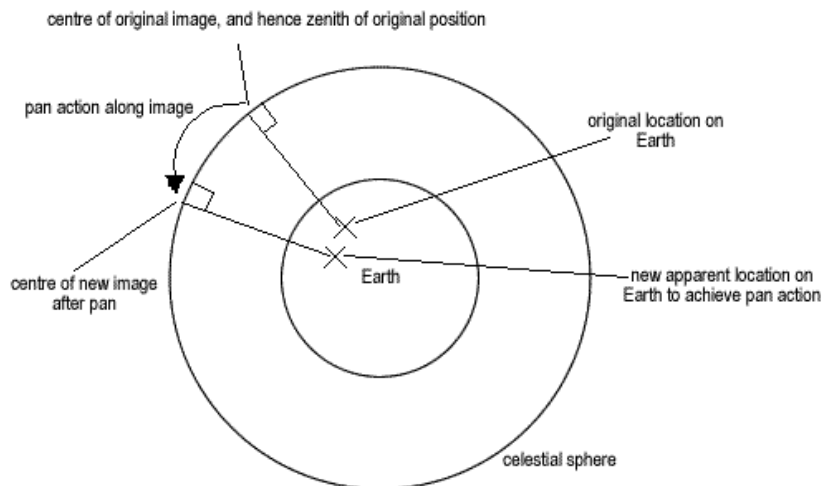


Figure 24: Panning concept

This new apparent location on Earth is then used in the calculation when converting the coordinates from the equatorial to the horizontal system to obtain the panned image.

This panning solution is then combined with the zooming, as shown in figure 25, in order to achieve the final working system.

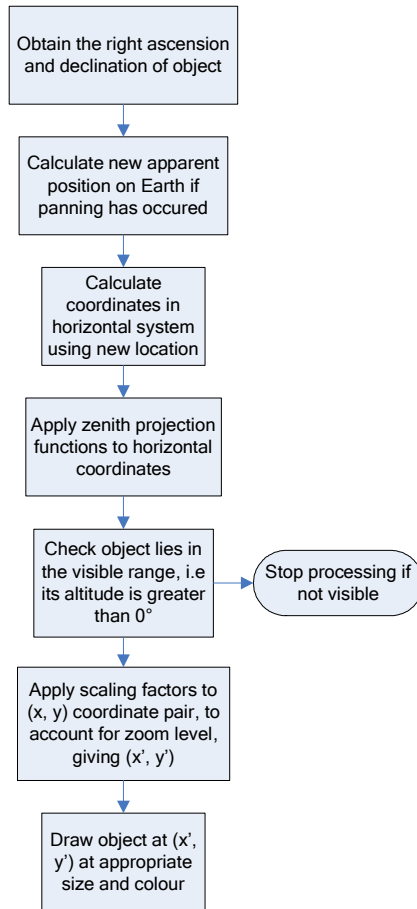


Figure 25: Tasks required to draw a celestial object within a zooming/panning system

7. Evaluation

The output of this system is an image and therefore requires manual analysis by studying the images generated to evaluate the system's results.

7.1 Correctness/Comparisons

The best way to check for correctness in this system has been to make direct comparisons with other existing solutions and check the positioning of the planets, and key constellations visible.

Test setup:

- Location: London, 52.5° N 0° E
- Time: 8.22am, 6th June 2004
- Magnification: 1x

Compared with:

- YourSky
- Sky View Café
-

Results:

The output of the final solution is shown in figure 26 and the images generated by YourSky and Sky View Café in figure 27 and 28 respectively.

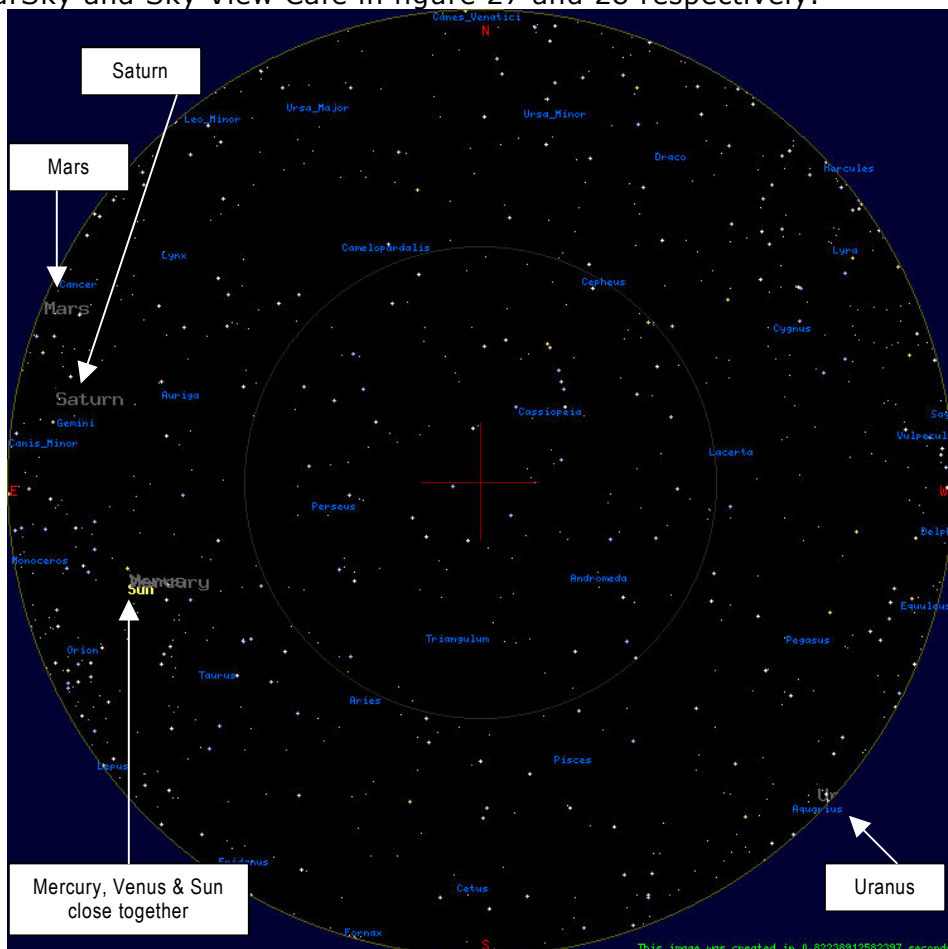


Figure 26: Sky generated by final system

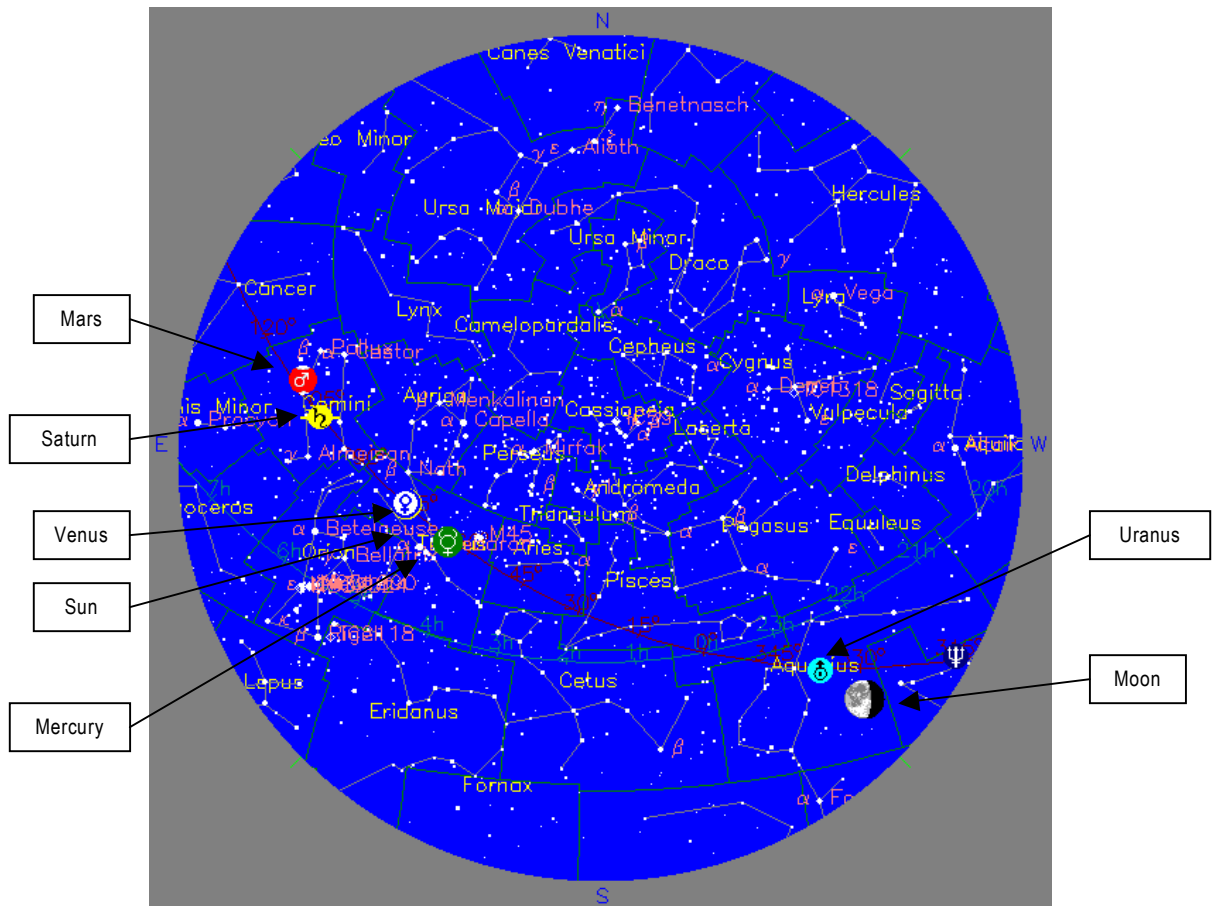


Figure 27: Sky generated by YourSky

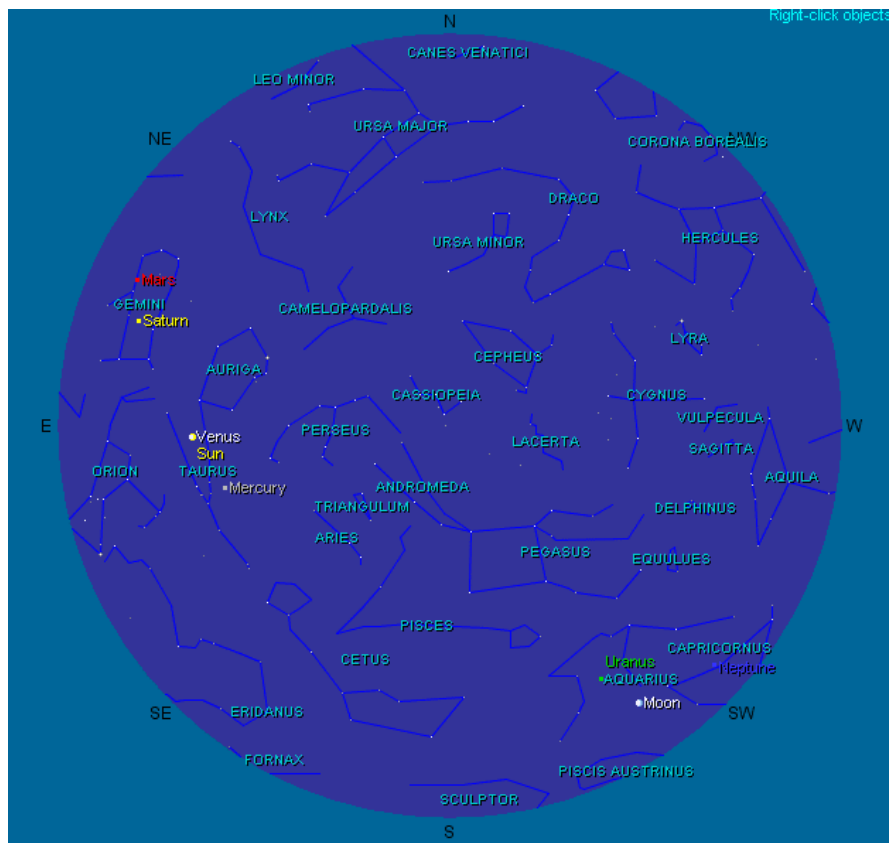


Figure 28: Sky generated by Sky View Cafe

7.1.1 Analysis

Constellations

The constellations in all three results lie in the same position, however the visible stars within the constellation do vary. The reason for this lies within the choice of the star catalogue and how the filtering weaker magnitude stars was done. Unfortunately due to a lack of lines on the final system, the constellations can be hard quite hard to identify and even with the labels it can be difficult.

Stars

Comparison of the position of the stars is difficult to judge and simply looking for similar areas of stars was the best way this could be done. Again the use of different catalogues and filtering techniques led to slightly different star maps.

Solar System objects

Sky View Café and YourSky produce identical results to each other, but differ to the implementation discussed in this report. The positions of the planets relative to each other are similar, however the absolute positioning differs. This can be caused by many reasons; however I feel it's due to slightly different implementations of the zenith projection and that my solution does not perform corrections at the edges of the projection.

Overall Clarity

Clarity is an important factor in such a system, where the only output seen is that of an image. One can judge a system's clarity quickly just by initial impressions of what they see. From the screenshots above it can easily be seen that YourSky is the most difficult to interpret, with an overload in information. The text size is too large and much overlapping occurs between names. My implemented system however tries to minimise the cluttering and takes a similar approach to that of Sky View Café by showing minimum information by default and then allowing the user to enable more options as desired.

Performance

Performance in such a system can be evaluated as the time taken to draw the image, however there is no accurate way of measuring this for existing solutions. All my PHP scripts have been timed internally and the time taken to complete the scripts is displayed in the bottom right of every image generated. For the setup outlined at the start of this section, the total time taken was 0.8 seconds using the bright star catalogue, which can be considered very fast for a web application. However using the All-sky catalogue and querying a database of approximately 2.5 million stars, increased this time to 4.3 seconds, which can cause a noticeably lag for users. The other existing solutions provided no noticeably lag and appeared to perform well in all circumstances.

8. Conclusion

The main aspects of this project have been covered by this report, detailing:

- The necessary background information required in implementing a planetarium on the web.
- A consideration of the current web technologies available today.
- Insight into the early decisions made during the implementation, and the reasoning behind their choice.
- A set of objectives to explore and meet.
- An overview of the system implemented and justification of its design
- A look into the core aspects of the system; client-side, server-side and backend database.
- The details of complications that arose during the development and how they were overcome or minimised.
- Finally an evaluation of the system of the developed and comparisons with existing solutions.

Much time has been spent working in developing this system and doing the astrophysics background research, and as a result I have gained much knowledge by completing it.

I have developed a good understanding of our solar system and the stars that lie beyond it, much of which had been unknown to me. The area of astronomy and astrophysics is a huge field and this project has only taken a tiny section from it and applied the available web technologies in creating in an application, that makes it accessible to a wide audience with no prior understanding of the field.

8.1 Future Work

Due to time restrictions, several sacrifices had to be made in order to make a functional system.

The method outlined in the background for calculating the Moon's position was only accurate to 4-6 arc-minutes and as a result certain astronomical events, such as solar eclipses, could not be seen to occur. Even slight errors would have caused the moon not to lie over the Sun, and this can be considered rather poor, since many people would use such a system in helping to identify positions of important events, to aid their sky gazing activities. It would definitely be worth investing time in implementing far more accurate algorithms, in order to calculate the positions of the Moon and even other bodies such as the planets, comets and man-made satellites.

The all-sky catalogue, the largest database tested with the system, containing 2.5 million stars, caused an obvious performance issue, causing a response time of anything up to 10 seconds for every user request. The fault was eventually identified as being caused by the queries sent to the all-sky table. The entire catalogue was entered into a single table, and a single query to obtain a list of stars at a specific magnitude and approximate position would take up to 10 seconds to execute. An improvement over this system would have been breaking the table into smaller tables, and spreading the data across them in an efficient manner that would increase the performance of the query's execution.

Another final improvement to increase the visual value of the application would be to implement correct constellation lines, connecting the appropriate stars. At the moment the image is rather plain and key constellation are not easily identifiable.

8.2 Summary

Many of these suggested improvements could easily be implemented under the current system architecture and technology used. As with every project, time is always an issue and the primary aim was to provide a final system with working functionality, which was achieved.

9. Bibliography

9.1 Background

- [B1] Curious About Astronomy – Stargazing?
<http://curious.astro.cornell.edu/stargazing.php>
- [B2] Basics of Positional Astronomy and Ephemerides
<http://www.jgiesen.de/SME/details/basics/index.htm>
- [B3] Computing planetary positions – a tutorial with worked examples
<http://www.stjarnhimlen.se/comp/tutorial.html>
- [B4] Coordinate conversion: celestial to horizon
<http://home.att.net/~srschmitt/celestial2horizon.html>
- [B5] Coordinate systems in Celestial Navigation
<http://www.jimthompson.net/boating/CelestialNav/CelestNotes/Coordinates.htm>
- [B6] US Naval Observatory – Data Services
<http://aa.usno.navy.mil/data/>
- [B7] Coordinate Systems in Astronomy
<http://www.krysstal.com/coordsystems.html>
- [B8] Answers.com – Mercator Projection
<http://www.answers.com/topic/mercator-projection>
- [B9] Orbits and Properties of the planet
<http://vathena.arc.nasa.gov/curric/space/planets/planorbi.html>
- [B10] Centre de Donnees astronomiques de Strasbourg
<http://cdsweb.u-strasbg.fr/>
- [B13] Projection for Navigators and Radio Operators
<http://www.progonos.com/furuti/MapProj/Normal/ProjNav/projNav.html#AzimutHalEquidistant>
- [B14] Peter Duffett-Smith
Practical Astronomy with Your Calculator, Third Edition
- [B15] A. E. Roy and D. Clarke
Astronomy – Principles and Practice, Third Edition
- [B16] About: Discover Secrets of Parallels and Meridians
<http://geography.about.com/cs/latitudelongitude/a/latlong.htm>
- [B17] What Colour Are the Stars?
<http://www.vendian.org/mncharity/dir3/starcolor/>
- [B18] The Geographer Craft – Map Projections
http://www.colorado.edu/geography/gcraft/notes/mapproj/mapproj_f.html
- [B19] Wikipedia – Mercator Projection
http://en.wikipedia.org/wiki/Mercator_projection
- [B20] All-sky Compiled Catalogue
(Kharchenko 2001)
<http://cdsweb.u-strasbg.fr/viz-bin/Cat?I/280A>
- [B21] The Hipparcos Catalogue
<http://vizier.u-strasbg.fr/viz-bin/Cat?I/239>

9.2 Technical

- [T1] PHP:Hypertext Preprocessor
<http://www.php.net>
- [T2] Potable Network Graphics
<http://www.libpng.org/pub/png/>
- [T3] PostgreSQL – Open source database

<http://www.postgresql.org/>

[T4] CSS School – W3S
<http://www.w3schools.com/css/default.asp>

[T5] GD Graphics Library
<http://www.boutell.com/gd/>

A1. Appendix

All data for epoch 1990 January 0.0 [B14 p105]

A1.1 Sun's apparent orbit at epoch

Ecliptic longitude at epoch	279.403303 degress
Ecliptic longitude of perigee	282.768422 degrees
Eccentricity of orbit	0.016713
Semi-major axis	1.1495985×10^8 km
Angular Diameter	0.533128 degrees

A1.2 Elements of the Moon's orbit at epoch

Mean longitude at the epoch	318.351648
Mean longitude of the perigee at the epoch	36.340410
Mean longitude of the node at the epoch	318.510107
Inclination of the Moon's orbit	5.145396
Eccentricity of the orbit	0.054900
Semi-major axis of the orbit.	
Angular size at apogee	
Parallax at apogee	

A1.2 Elements of the planetary orbits

Planet Name	Period (tropical years)	Longitude at epoch (degrees)	Longitude of the perihelion (degrees)	Eccentricity of the orbit	Semi-major axis of the orbit (AU)	Inclination of the orbit (degrees)	Longitude of the ascending node (degrees)	Angular diameter at 1AU (arcsec)	Visual magnitude at 1AU
Mercury	0.240852	60.750646	77.299833	0.205633	0.387099	7.004540	48.212740	6.74	-0.42
Venus	0.615211	88.455855	131.430236	0.006778	0.723332	3.394535	76.589820	16.92	-4.40
Earth	1.0004	99.403308	102.768413	0.016778	1.00000	NA	NA	NA	NA
Mars	1.863075	240.739474	335.874407	0.093396	1.523688	1.849736	49.480308	9.36	-1.53
Jupiter	11.863075	90.638185	14.170747	0.048482	5.202561	1.303613	100.352142	196.74	-9.40
Saturn	29.471362	287.690033	92.861407	0.055581	9.554747	2.4888980	113.576139	165.6	-8.88
Uranus	84.039492	271.063148	172.884833	0.046321	19.21814	0.773059	73.926961	65.8	-7.19
Neptune	164.79246	282.349556	48.009003	0.009003	30.109570	1.770646	131.670599	62.2	-6.87
Pluto	246.77027	221.4127	224.133	0.24624	39.3414	1.770646	110.144	8.2	-1.0

1AU = 149.6 x 10⁶ km