

**Christian Lenz's**

# **STC-Scripting Guide for Celestia**

<http://www.shatters.net/celestia/>

**Version 1.2 (English)**

**Rev. 1**

**© by Christian "Guckytos" Lenz, January 2006**

<http://home.arcor.de/christianlenz/>

## Table of Contents

Introduction.....	3
STC-file layout.....	4
Basic STC parameters for a normal star.....	5
Elements of the STC-file.....	5
Object designator.....	5
OrbitBarycenter.....	6
RA.....	7
Dec.....	9
Distance.....	9
SpectralType.....	10
Spectral Classes known by Celestia.....	10
Spectral class.....	10
Hertzsprung-Russell-Diagram (HRD).....	11
Luminosity Classes.....	12
Rare and unusual spectral classes.....	13
Absolute Magnitude or Apparent Magnitude.....	16
Radius.....	17
EllipticalOrbit.....	17
SemiAxes.....	22
Mesh.....	23
Texture.....	24
Rotational Elements.....	25
Credits.....	27

## Introduction

One of the features of Celestia is the ability to create your own stars or stellar systems.

To create a solar system two catalog files with the necessary definitions are needed.

- A **STar Catalog** (STC), which contains the characteristics and position of the object that Celestia has to create. A STC-file can be used to describe not only a star, as its name indicates, but also a barycenter for orbiting objects. And around these two classes of objects other stars, barycenters or objects, which are defined in SSC files, can be put into orbit.
- A **Solar System Catalog** (SSC), which contains the characteristics and orbital details of planets and other objects around the star.

This guide takes a detailed look at the STC-file that is needed to tell Celestia the position and further characteristics of the star. If you want to know more about creating planetary systems and thus SSC-files then take a look at the excellent guide from Ulrich Dickmann and Bob Hegwood (English version) or Ulrich Dickmann (longer German version). These can be either found on the Celestia Motherlode page <http://www.celestiamotherlode.net> in section Resources/Documentation or at <http://www.celestia.de.vu> (German version) or <http://www.bobhegwood.com> (shortened English version).

A STC-file can either contain only a single definition for one star/barycenter or the definitions for a whole lot of stars/barycenters. So with Celestia version 1.4.0 it is now possible to create binary systems or star systems with even more suns with only a STC-file. If you are still using an earlier version of Celestia you will need a STC- **and** a SSC-file for this purpose. But it will never look as good as in version 1.4.0.

The STC-file can be created with every text editor, and then saved as plain text. There are only two requirements that have to be met by the STC-file:

1. Every star definition has to be complete and correct in its syntax: Otherwise Celestia will ignore this and all following definitions.
2. Every STC-file **must** have an extra blank line at its end or it won't function.

## STC-file layout

Warning, the here described features will **not** work with versions lower then 1.4.0!!  
For the development of stars with Celestia versions up to 1.3.2 see the next chapter  
“Basic STC parameters for a normal star”.

The following example shows the layout of a STC-file.

```
#####
# Comment
#####
# HIP 500000
Object designator      # Object class, HIP-number and/or object name
{
    OrbitBarycenter "Name" # Place a star/barycenter into orbit around "Name"
    RA 100.00             # Right Ascension in degrees
    Dec 45.00             # Declination in degrees
    Distance 1000         # in Light Years
    SpectralType "A0V"    # Spectral type
    AbsMag 30             # Absolute Magnitude (also AppMag possible)
    Radius 1250000        # Radius of the star in km

    EllipticalOrbit
    {
        Period          90.00      # Duration of orbit in years
        SemiMajorAxis    100.00    # Radius of orbit in AU
        Eccentricity      0.1      # Defines the shape of the orbit
        Inclination       80.00    # in degrees
        AscendingNode     70.00    # in degrees
        ArgOfPericenter   10.00    # in degrees
        MeanAnomaly       150.00   # in degrees
        Epoch              # Julian Date or Universal Time
    }

    SemiAxes [1 0.85 1]    # defines the shape of a star along 3 main axes
    Mesh "MyMesh"          # replaces the normal mesh for a star
    Texture "My Texture"   # replaces the normally used star texture

    RotationPeriod        480      # in hours
    Obliquity              3.3947   # in degrees
    EquatorAscendingNode   325      # in degrees
    RotationOffset         280.5    # in degrees
    PrecessionRate         48.98    # in rad/days
    Orientation            [180 1 0 0]

}
<Extra blank line>
```

Celestia ignores everything on a line after a #, so these can be used for comments.  
So the first four lines in the example are just comments. But those comments can be  
quite useful, if there are more definitions in one file and if they are done properly and

you want to know at a glance what is in the file. Especially if a not very descriptive name has been given to the file, or it was created a long time ago.

## Basic STC parameters for a normal star

If you say “Whoaaa, what a lot of parameters that I need to understand in order to create just a normal star!”, don’t despair. All you need if you just want to place a star at a certain position is shown in the STC-layout below. But this works only for a star that is **not** orbiting anything else. But of course you can place anything into an orbit around that star (\*g\*). You don’t need to specify any object type to create a star, if nothing stands there Celestia assumes you want a star there. You can give a HIP number to your star and/or a name, but one of the two is a must.

```
#####  
# Comment  
#####  
# HIP 500000  
HIP number and/or “Star name” # Hipparcos number and/or assigned star name  
{  
    RA 100.00      # Right Ascension  
    Dec 45.00      # Declination  
    Distance 1000   # in Light Years  
    SpectralType "type"# Spectral type  
    AppMag 15       # Apparent Magnitude  
}  
<Extra blank line>
```

This basic layout has all the commands in it that Celestia knows up to version 1.3.2. So if you are using a version before 1.4.0, all you have to care about are those commands above. One more thing: If you are using an older version (1.3.2 or earlier) you **must** give HIP number **and** “Star name”.

## Elements of the STC-file

### Object designator

In the new version are now three optional fields used to introduce an STC object. If all three fields are going to be used OR if you want to place a barycenter, the following order of fields has to be used:

1. Type of Object
2. HIP number
3. Name of object

**Type of object** (optional) = either **Barycenter** or **Star**. A barycenter is the center of gravity between two or more objects, around which these objects have orbits. If this field is omitted, Celestia assumes that you want to place a star.

**Important!** The **HIP number** and “**Name of object**” field are both optional, but at least a HIP number **or** a Star name is required for Celestia to be able to create and place that object!

**HIP number** (optional) = Celestia uses "Hipparcos numbers", short HIP, to keep track of stars. Hipparcos numbers are numbers assigned by the astronomers who used the Hipparcos satellite to measure the distances to many stars. Imaginary stars have to be given fake Hipparcos numbers in Celestia. Usually people use numbers between 300000 and 600000, to avoid conflicts with existing real stars that are in the database of Celestia.

If a HIP number for a new object is used, that has the same HIP number as an already existing star or barycenter, that object is replaced by the new object. Barycenters can replace Stars.

**"Name of object"** (optional) = The name for the newly created object. Be careful with your names, because if you choose a name that has already been given to a star that Celestia knows, the results can be confusing, i.e. the "new" star could be created at the position of the star from Celestias own database and not its wanted position.

It is possible to assign more than one name to a star, by separating the names with a colon in the Star name field.

In Celestia you can now go to your star by using either the Enter/Name/Enter method, entering any of the names of the definition or entering the HIP number (in the form "HIP 450000"). If you choose to not give your star or barycenter a name then the only method to visit it is to enter the HIP number.

Example 1: **Barycenter 450000 "My Barycenter:My System"**

Defines a barycenter with the HIP number 450000 and the names “My Barycenter” and “My System”. Both names lead to the same system.

Example 2: **"First star:The first:Beginning"**

Defines a star with the names “First star”, “The first” and “Beginning” and no HIP number.

## OrbitBarycenter

**OrbitBarycenter** (optional) = The location of a star or barycenter around another star can be specified by either entering RA, Dec and Distance for this object as usual or by using the keyword OrbitBarycenter to specify the name of a previously defined barycenter or star around which the new object should orbit. A star or barycenter can orbit around another star or barycenter.

If the OrbitBarycenter option is used only the **SpectralType** and **AbsMag** have to be defined for a new star. If a barycenter should be placed into orbit around another object, see Example 6.

For the next example the star “**First Star**”, which was defined above at a specific RA, Dec and Distance, will get a companion in the examples below. There are different ways to do this, so all are presented:

Example 3: "My Second Star"

```
{
  RA 299.0           # in fractional degrees
  Dec -10.5          # in fractional degrees
  Distance 30        # in light years
  SpectralType "G2V"
  AbsMag 7
}
```

Example 4: "My Second Star"

```
{
  OrbitBarycenter "First Star"
  SpectralType "G2V"
  AbsMag 7
}
```

Examples 3 and 4 create the same star "My Second Star" with the same characteristics at the same position, which is the position of "First Star". Which method you want to use is more or less your decision. Specifying RA, Dec and Distance is slightly more efficient than OrbitBarycenter, but only at load time, when Celestia has to look up the position of the Barycenter.

But if **OrbitBarycenter** is used, then **EllipticalOrbit** for this object must be specified too. The reason is, that you want to have one object orbiting around another and the Celestia than must know the specifications of that orbit.

Example 5: **BaryCenter** "First Barycenter"

```
{
  OrbitBarycenter "First Star"
}
```

In Example 5 a barycenter has been created named "First Barycenter" at the same position as the second star in the examples above. This barycenter orbits around "First Star". As easily you could create two or more barycenters.

Example 6: **BaryCenter** "My Barycenter"

```
{
  OrbitBarycenter "First Barycenter"
}
```

In Example 6 another barycenter has been defined, that is put in orbit around "First Barycenter". If you add up the examples, you have now 2 suns, that orbit each other and two barycenters that orbit around "First Star" and also around each other. For all those objects the **EllipticalOrbit** descriptions have to be defined!

## RA

**RA** = Right Ascension, the angle in degrees in the equatorial plane of the earth, measured from the Vernal Equinox where the star is located. The angle runs counter-clockwise. A bit further down is a figure with an explanation of the Vernal Equinox.

Celestia uses degrees with decimals for the RA, instead of the in astronomy normally used format of hours (<sup>h</sup>), minutes (<sup>m</sup>) and seconds (<sup>s</sup>). Therefore perhaps a conversion of the angles has to be undertaken, if a star is placed.

$1^h = 60^m$  and  $1^m = 60^s$ , a full circle of 360 degrees represents  $24^h$ , so 1 hour is the equivalent of 15 degrees.

To convert any given RA from time to degrees use the following equations:

$$\text{RA (Time)} = \text{hours} + \text{minutes}/(60 \text{ min/h}) + \text{seconds}/(3600 \text{ sec/h})$$

$$\text{RA (Degree)} = \text{RA (Time)} * 15 \text{ degrees/hour}$$

Example: **RA = 11<sup>h</sup> 27<sup>m</sup> 35.9<sup>s</sup>**

$$\text{RA (Time)} = 11 + 27/60 + 35.9/3600 \text{ hours} = 11.45997222 \text{ hours}$$

$$\begin{aligned} \text{RA (Degree)} &= 11.45997222 \text{ hours} * 15 \text{ degrees/hour} = \\ &171.8995833 \text{ degrees} \end{aligned}$$

So the entry for the STC-file could look like this:

Example: **RA 171.8995833**

Now how many decimals do you need for the star? That depends on how accurate you want to position it and that depends also on how far away the star should be. Remember, that if you increase the distance you also increase the distance between two neighboring points on the arc. That is something you have to decide for yourself. I use normally seven decimals, because that is accurate enough for my purposes, and will probably be also true for you too, but use whatever you are happy with.

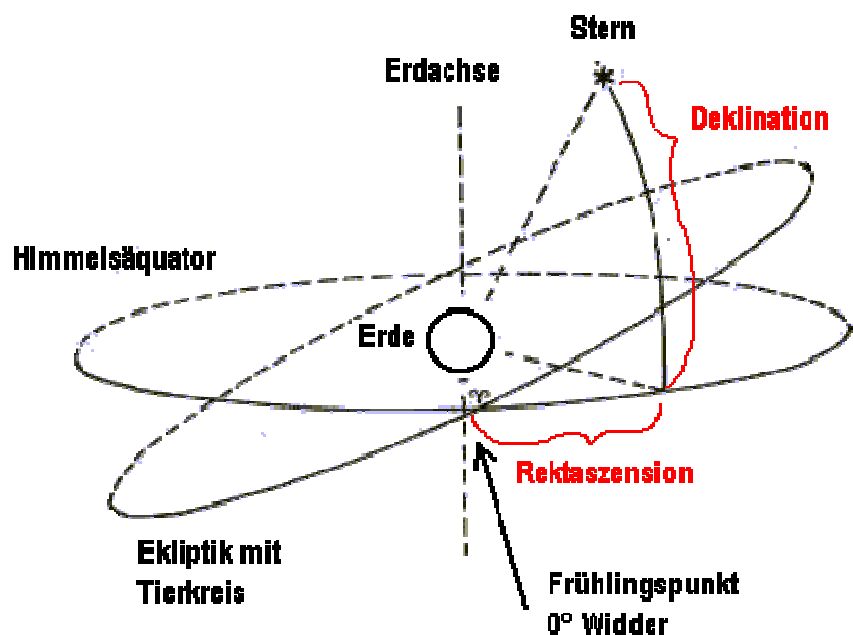


Fig. 1: Equatorial coordinate system



This diagram has been taken from the German-speaking version of the SSC-Guide by Ulrich Dickmann. So here are some translations of the words:

Stern = star  
Erdachse = rotation axis of the earth  
Himmelsäquator = celestial equator  
Erde = earth  
Frühlingspunkt = vernal equinox  
Rektaszension = Right Ascension, RA  
Deklination = Declination, Dec

Through the movement of the earth on its orbit around the sun the position of the sun seems to change from day to day for an observer, who is on a fixed position on earth. On the first day of spring (21<sup>st</sup> of March) the sun crosses the celestial equator (the into infinity projected equator of the earth) on her apparent orbit. This point of intersection between the sun's orbit, the ecliptic and the celestial equator is called **Vernal Equinox**. Another possible name would be "ascending node of the Sun's orbit"<sup>1</sup>.

## Dec

**Dec** = the Declination is the angle between the connecting line between earth and star and the celestial equator. It defines how far above or below the equatorial plane the star is positioned. Celestia uses degrees with decimals for this purpose. In astronomy on the other hand the declination is given in the format of degrees, arc minutes and arc seconds. So the conversion to degrees works a bit different from that of the RA.

1 degree = 60' (arc minutes) and 1' = 60" (arc seconds)

Dec = degrees + arc minutes/(60 deg'') + arc seconds/(3600 deg'')

Example: **Dec = -11 degrees 39' 45"**

Dec = -(11+39/60+45/3600) = -11.6625 degrees

So the entry for the STC-file could look like this:

Example: **Dec -11.6625**

## Distance

**Distance** = The distance of the star from the Earth is given in light years. Celestia is only able to create stars up to a distance of 16,000 light years. If a greater value is used, the coordinate point will be created, but no star will be positioned. There is also another measure for distance in astronomy; the parsec (parallactic second). The conversion is quite easy:

1 parsec (pc) = 3.2616 light years (ly)

---

<sup>1</sup> Taken from the manual for .SSC-files by Ulrich Dickmann and Bob Hegwood

## SpectralType

**SpectralType** “type” = defines the spectral color and some other characteristics of the star. This topic is a bit more complex and will be explained in detail.

Example: The Sun = **SpectralType** “G2V”

What does that mean and what spectral types do exist and which types does Celestia know? In short, the spectral type is a combined expression of **spectral** and **luminosity class**. Our sun for example is a member of the spectral class “G”, subcategory “2” and in the luminosity class “V”.

If only the spectral class is given in the definition, Celestia will automatically create the star as a main sequence star.

## Spectral Classes known by Celestia

Celestia is able to create stars from the following spectral classes of stars.

The Morgan-Keenan spectral classes: O, B, A, F, G, K, M

The following rare or unusual classes: L, T, R, S, N, C, WC, WN,  
D (white dwarf)  
Q (Neutron star)  
X (Black hole)

## Spectral class

The following three chapters are meant to give an overview about the question, “What is a spectral class and what does it tell about a star and its properties?”.

It might be useful for beginners, but if you already know what types of stars you want to create, you can skip them.

Stars are obviously emitting light, otherwise we wouldn’t be able to see them, but they do not only have different luminosities, but also different colors. The color of a star depends on its composition and its surface temperature, similar to glowing iron. When iron is heated it starts first to glow deep red, going through yellow and finally being bright white, at the same time its luminosity also increases. The same is in principle true for stars; the luminosity only depends on their surface temperature (color) and their emitting area (or rather their diameter). A cool red giant is brighter than a normal red star.

If the light of a star is run through a prism, it is divided into its spectral colors. The diagrams from this will show that a red (cool) star doesn’t show much of blue light emissions, whereas a white (hot) star shows a lot of it. And you will also get “finger-prints” of the atoms of which the surface of the star, and the star itself, is composed as lots of small dark lines, where the atoms absorb light with characteristic wavelengths. These lines exist because atoms are able to absorb light with characteristic wavelengths that correspond with certain characteristic energy levels. Which atoms are able to absorb which wavelengths is exactly defined by the composition of the atom. Therefore it is possible to derive from the spectral lines not only the temperature of the star, but also the composition of the star itself.

These patterns of the spectral lines are closely coupled to the temperature of the star. From these spectral lines the spectral classes have been derived. In the beginning it was only differentiated between a few (seven) distinctive spectral classes that are described with an alphabetic letter, but with evolving observation technology and understanding of stellar properties new classes have been added. The seven main classes range from O, the hottest stars, to M, the coolest stars. There are a lot of different mnemonics for the sequence from hot to cool, one is:

**“O Be A Fine Girl And Kiss Me”.**

These spectral classes are further divided into subclasses by adding a numerical value between 0 and 9 to the letter. The higher the number is, the cooler is the star. So a M9 star is the coolest star in this classification system.

Type	Color	Approximate Temperature
O	Blue	> 30,000 K
B	Blue to White	11,000 - 30,000 K
A	White	7,500 - 11,000
F	White to Yellow	6,000 - 7,500
G	Yellow	5,000 - 6,000
K	Yellow to Orange	3,500 - 5,000
M	Red	< 3,500

### Hertzsprung-Russell-Diagram (HRD)

When Hertzsprung and Russell entered the data of a lot of stars into a diagram, which links their luminosity (i.e. absolute magnitude) with their temperature respectively spectral class, they found out that the stars in the diagram weren't evenly distributed. About 90 percent of the stars lie on a line that stretches in a slight S-form from the upper left to the lower right of the HRD. This line is the so-called Main-Sequence and the stars in it are called Main-Sequence Stars. The stars that lie above the Main-Sequence are called giants and they are arranged in lines that branch out from the Main-Sequence. Those stars are relatively cool, but have a bigger diameter than main sequence stars and therefore a higher luminosity (red giants). Only very few stars are located below the Main-Sequence, those stars are called white dwarfs.

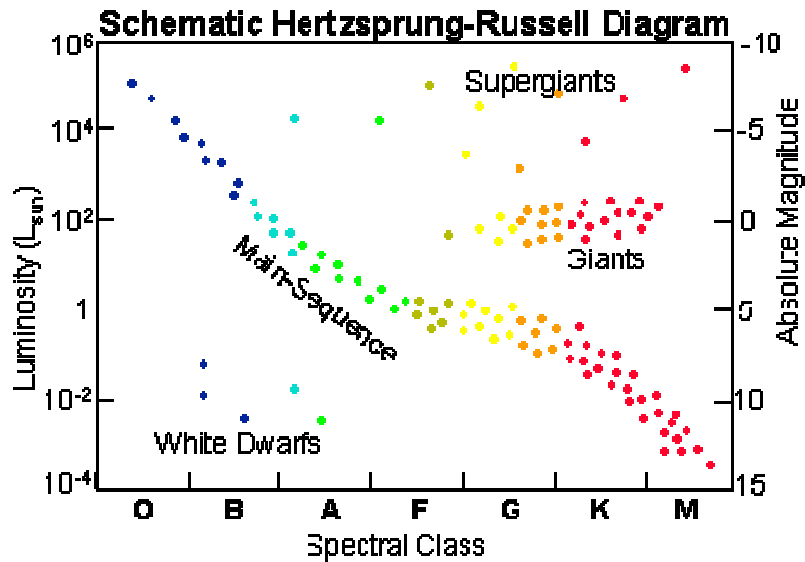


Fig. 2: HRD (taken from Astronomy 101/103, Terry Herter, Cornell University)

### Luminosity Classes

Luminosity is the total energy that a star radiates per second. A luminosity class is designated by a Roman numeral, in case of giants eventually enhanced with a letter or a letter-number combination. This luminosity class is appended to the spectral type. These classes correlate with the age of the star and its size: Larger stars are brighter, even if all other things (spectral class, distance etc.) are equal.

There are six main classes recognized, seven if the white dwarfs are included. In case of the giants a finer division in the class is made.

It is possible to make a HRD where only the luminosity is plotted over the spectral classes (see Fig. 3). It can be clearly seen that all Main-Sequence Stars have the luminosity class V, if this figure is compared to Fig. 2. Brighter stars, like the giants have a lower roman number, than dimmer stars.

If no luminosity class is chosen, Celestia will automatically define the star as a Main Sequence star.

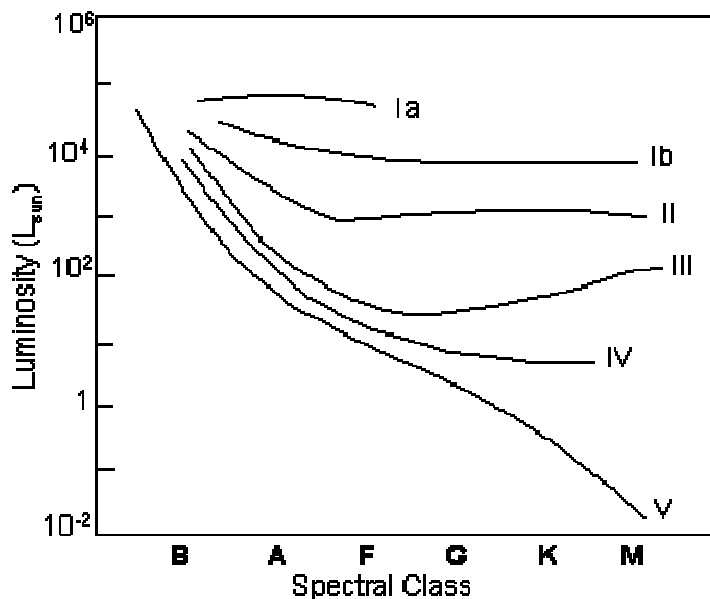


Fig. 3: Luminosity classes (Astronomy 101/103, Terry Herter, Cornell University)

Luminosity Class	Star Size
Ia0	Extreme, luminous supergiants
Ia	Luminous supergiants
Ib	Less luminous supergiants
II	Bright giants
III	Normal giants
IV	Sub-giants
V	Main sequence (dwarfs)
VI, sd	Sub-dwarfs
D	White dwarfs (have their own spectral class in Celestia)

## Rare and unusual spectral classes

For rare and unusual stars the following spectral classes are recognized within Celestia.

- L:** Stars with insufficient mass to run the regular hydrogen fusion process.  
 Temperatures 1,500 - 2,000 K  
 Class L stars get their designation from the lithium present in their core. Any lithium would be destroyed in ongoing nuclear reactions in regular stars, which indicates that these objects have no ongoing fusion processes.  
 They are a very dark red in color and brightest in infrared. Their gas is cool enough to allow metal hydrides and alkali metals to be prominent in the spectrum.
- T:** T Tauri stars, very young and low density stars.  
 Temperatures ~1,000 K  
 Class T stars are very young and low density stars often found in the interstellar clouds they were born in. The gravitational contraction is still in process and they are barely big enough to be stars. T Tauri stars are in a transitional phase between being proto-stars and stars arriving at the main sequence. On a Hertzsprung-Russell Diagram, they plot above the main sequence.  
 They appear black, emitting little or no visible light but being strongest in infrared. Complex molecules can form, evidenced by the strong methane lines in their spectra.
- R:** Formerly a class on its own, representing the carbon star equivalent of Class K stars. Has been remapped into the new Carbon star class C.  
 Very rare

**N:** Formerly a class on its own representing the carbon star equivalent of Class M stars. Has been remapped into the new Carbon star class C.

Class R and N stars are giant stars that appear to be G to M type stars, but also show significant excess spectral features of carbon compounds. They are often referred to as "carbon stars". The presence of carbon compounds in these stars will tend to absorb the blue portion of the spectrum, giving R and N type giants a distinctive red color. R stars are those with hotter surfaces which otherwise more closely resemble K type stars.

These two classifications have been remapped into a unified carbon class C.

**S:** Stars between class M stars and the carbon stars, but with zirconium oxide replacing the regular titanium oxide.

S type stars show heavier elements such as zirconium, yttrium, and barium instead of the usual lines of titanium, scandium, and vanadium oxides characteristic of M type giants.

A significant fraction of all S type stars are variable.

**C:** Carbon stars, overlap with late G, K and M stars in most respects but are distinguished by compositional differences, typically an unusually high concentration of carbon. Carbon stars are typically long period variables.

The former classes of R and N stars have been remapped into the C class. Type R is the approximate carbon analogue of the K stars; N is the approximate carbon analogue of the M stars.

The approximate equivalence between carbon stars (Harvard and Keenan & Morgan classifications) and normal stars is shown in the following table. There is good evidence that the N stars correspond to M classes later than given here. The C-type responds to both carbon abundance and temperature and consequently there is a large and essentially unknown temperature range for each carbon subclass.

Harvard Type	Keenan & Morgan Type	Equivalent Normal Star
	C0	G4 – G6
R0	C1	G7 – G8
	C2	G9 – K0
R3	C3	K1 – K2
R5		
	C4	K3 – K4
R8	C5	K5 – M0
	C6	M1 – M2
N1		
N2	C7	M3 – M4
	C8	
N3		
	C9	

**W:** The main class W represents the superluminous Wolf-Rayet stars, being notably different since they have mostly helium instead of hydrogen.  
Temperatures up to 70,000 K

They are thought to be dying supergiants with their hydrogen layer blown away by hot stellar winds caused by their high temperatures, thereby directly exposing their hot helium core.

Wolf-Rayet stars are similar to O type stars, but have broad emission lines of hydrogen and ionized helium, carbon, nitrogen, and oxygen with very few absorption lines. Current theory holds that these stars exist in binary systems where the companion star has stripped away the Wolf-Rayet star's outer layers.

**Subclasses:**

**WC:** the hottest though least luminous type; oldest (most evolved) stars of the W-R sequence

Hydrogen absent, surface carbon presence around 40%, oxygen prominent, nitrogen absent; least massive

**WN:** coolest and brightest of the W-R stars

Most massive; large radius; the youngest (least evolved) stars of the sequence; thought to comprise a He burning core and H burning shell

**D:** White dwarves

White dwarfs are the cooling naked cores of the most highly evolved stars which have mostly consumed their fusionable elements and undergone gravitational collapse. None is bright enough to be observed with the naked eye.

White dwarfs are not classifiable by ordinary rules. For example, more massive white dwarfs are usually less luminous because their higher gravity contracts the stars more, reducing the surface area. Color does not correlate with temperature as each type spreads out across the whole temperature sequence from early O-type to M equivalents.

Being insufficiently massive to collapse into a neutron star, further collapse is prevented by electron degeneracy pressure.

**Subclasses:**

**DA:** White dwarf A (Balmer lines, no He I or metals); Temperature between 37,500 and 100,000 K

**DB:** White dwarf B (He I lines, no H or metals)

**DC:** White dwarf C, continuous spectrum; Temperature 5500K

**DO:** White dwarf O, He II strong, He I or H; Temperature 70,000 K

**DQ:** White dwarf Q, carbon features

**DX:** White dwarf X, unidentified features; Temperature 10,000 K

**DZ:** White dwarf Z, metal lines only, no H or He; Temperature 8500 K

**Q:** Neutron stars, like the White Dwarfs an endpoint in stellar development. The original mass of the star is high enough to overcome the electron degeneracy pressure and to collapse protons and electrons into neutrons.

The star has a neutron core with densities of about  $10^{14}$  g/cm<sup>3</sup>. Radius is in the range of 8 to 20 km. Very short rotation periods in the range from milliseconds up to some seconds.

X: Black Hole, not to be recommended to use it, since Celestia assumes an infinite radius for it, which can lead to unstable behavior of the program.

## Absolute Magnitude or Apparent Magnitude

**AbsMag** = The **absolute magnitude (M)** of a star is a measure for the brightness it would have if it was at a standardized distance from the earth. It allows to compare the overall brightness of objects without regards to distance. In stellar astronomy, the standard distance is 10 parsecs (which is about 32.616 light years). The absolute magnitude of a star will not change with distance.

**AppMag** = The **apparent magnitude (m)** of a star is a measure for its apparent brightness, as seen from the earth as center of reference. That is, the amount of light received from the star, this is also true for any other object. As you travel to a star its apparent magnitude (m) will increase.

**Only one keyword** is necessary for a star. Both keywords can equally be used to describe the same star, but be sure which one you use with which value, or you might get stars that are way to dim or to bright!

Negative values of magnitude indicate brighter stars, while positive values indicate low luminosity. The dimmer an object appears, the higher its apparent magnitude. Note that apparent brightness is not equal to actual brightness — an extremely bright object may appear quite dim, if it is far away.

One thing to notice is that the magnitude scale is logarithmic: the relative brightness of two objects is determined by the difference of their magnitudes. The logarithmic nature of the scale is due to the fact of the human eye itself having a logarithmic response.

The modern system of magnitudes has assigned negative magnitudes to really bright objects. For example, Sirius, the brightest star of the celestial sphere, has an apparent magnitude of  $-1.44$  to  $-1.46$ . The modern scale includes the Moon and the Sun; the Moon has an apparent magnitude of  $-12.6$  and the Sun has an apparent magnitude of  $-26.7$ . The Hubble and Keck telescopes have located stars with magnitudes of  $+30$ .

The absolute magnitude (M) is related to the apparent magnitude (m) and true distance to the star (d in parsecs) by

$$M = m - 5 \log (d/10)$$

$$m = M + 5 \log (d/10)$$

Many stars visible to the naked eye have an absolute magnitude which is capable of casting shadows from a distance of 10 parsecs. Rigel ( $-7.0$ ), Deneb ( $-7.2$ ), Naos ( $-7.3$ ), and Betelgeuse ( $-5.6$ ).



A rough absolute magnitude for given a star can be taken out of the Hertzsprung-Russell-Diagram. From this, and the distance you want to place the star, you are able to calculate an approximate apparent magnitude for your star, so it won't look too bright or too dim.

Example: Rigel =   SpectralType "B8 I-a"  
                          Distance 772.91  
                          AbsMag -7.0

or   SpectralType "B8 I-a"  
      Distance 772.91  
      AppMag -0.126

**But don't use both magnitude definitions at the same time for the same star!**

Okay, in Celestia you will find a slightly different value for the apparent magnitude, because it uses another value for the absolute magnitude, but you got the idea, right?

## Radius

**Radius** (optional) = allows you to change the given (real) radius for this star to any value of your desire. It is entered in kilometers.

Example: **Radius** 1500000

This will create a star with a diameter of 3 million kilometers.

## EllipticalOrbit

The orbit of a Star or Barycenter is specified using the same **EllipticalOrbit** declarations that are used in SSC files. This elliptical orbit is relative to the object's location, which is specified using either an **RA**, **Dec** and **Distance** triplet or by an **OrbitBarycenter** command.

All definitions and pictures in this paragraph have been taken from the manual for .SSC-files by Ulrich Dickmann and Bob Hegwood and just slightly modified and trimmed down a bit. If more or better explanation is needed, please look them up in the mentioned manual.

**EllipticalOrbit** = The set of elements used to describe the orbit of an artificial satellite. The collection of elements listed under the EllipticalOrbit heading **MUST** be defined between a set of braces as shown below.

```
EllipticalOrbit
{
    Period           0.6152
    SemiMajorAxis    0.7233
    Eccentricity     0.0068
    Inclination       3.3947
```

```

AscendingNode 76.681
LongOfPericenter 131.533 (also: ArgofPericenter)
MeanLongitude 181.979 (also: MeanAnomaly)
Epoch
}

```

Orbital elements remain a mystery to most people. This is due, I think, first to the aversion many people (including Bob and me) have to thinking in three dimensions, and second to the horrible names the ancient astronomers gave these seven simple numbers and a few related concepts. To make matters worse, sometimes several different names are used to specify the same number. Vocabulary is the hardest part of celestial mechanics!

**Period** = The time it takes for an object to orbit its host. Periods for stars are defined in terms of years. The Period element **MUST** be defined.

**SemiMajorAxis** = A number which identifies half of the large axis of the object's orbit. Generally speaking, the SemiMajorAxis is also the average distance from a celestial body to its orbiting companion. The element is defined in terms of Astronomical Units (AU) for planets and now also for the distance between orbiting stars in Celestia

Note: One AU = approximately 150 Million Kilometers. The SemiMajorAxis element **MUST** be defined.



The major axis (marked as "Apsidenlinie") runs from "A" to the point "P". In that diagram, the orbit is a circle (not an ellipse), so the distance from the center to "A" is the same as from center to "P".

On elliptical orbits, the semi major axis is always the longer line, that points to "A" (apogee).

Bahnebene = Orbital plane

**SemiMajorAxis** is one-half the length (measured the long way) of the orbit ellipse.

**Eccentricity** = A number between 0.0 and 1.0 which defines the circularity of the object's orbit. A perfectly circular orbit would have an Eccentricity value which is equal to zero. A radically eccentric orbit would have an Eccentricity value of one. This element is optional, but should be used if you desire accuracy.

This one is simple. In the Keplerian orbit model, the satellite orbit is an ellipse. Eccentricity tells us the "shape" of the ellipse.

When eccentricity = 0, the ellipse is a circle.

When eccentricity is very **near** 1, the ellipse is very long and skinny.

**Inclination** = A number which defines the plane of the object's orbit in comparison to the plane of the terrestrial equator. This element is optional, but should be used for accuracy.



The orbit ellipse lies in a plane known as the orbital plane (see “Bahnebene”).

The orbital plane always goes through the center of the earth, but may be tilted at any angle relative to the equator.

Inclination is the angle between the orbital plane (red) and the equatorial plane (blue).

By convention, inclination is a number between 0 and 180 degrees.

Some vocabulary: Orbits with an inclination near 0 degrees are called equatorial orbits (because the satellite stays nearly over the equator). Orbits with inclination near 90 degrees are called polar (because the satellite orbits over the north and south poles). The intersection of the equatorial plane and the orbital plane is a line which is called the line of nodes. More about that later.

**AscendingNode** = The ascending node is the point of intersection of the orbit with the plane of the equator when a satellite travels from the southern hemisphere towards the northern hemisphere. Or, in other words: The ascending node is where the Orbit crosses the Ecliptic. This number (given in degrees) defines the right ascension of the ascending node in degrees. It determines the orientation of the axis of the nodes compared to the direction of reference (i.e. the Vernal Point).

So we take a closer look at this stuff:

Two numbers orient the orbital plane in space. The first number was Inclination. This is the second. After we've specified inclination, there are still an infinite number of orbital planes possible. The line of nodes can poke out the anywhere along the equator. If we specify where along the equator the line of nodes pokes out, we will have the orbital plane fully specified. The line of nodes pokes out two places, of course. We only need to specify one of them. One is called the ascending node (where the satellite crosses the equator going from south to north). The other is called the descending node (where the satellite crosses the equator going from north to south). By convention, we specify the location of the ascending node.

Now, the earth is spinning. This means that we can't use the common latitude/longitude coordinate system to specify where the line of nodes points. Instead, we use an astronomical coordinate system, known as the right ascension / declination coordinate system, which does not spin with the earth.

Right ascension is another fancy word for an angle, in this case, an angle measured in the equatorial plane from a reference point in the sky where right ascension is defined to be zero. Astronomers call this point the vernal equinox. Vernal equinox? What's that? We'll talk about it later...

Finally, "right ascension of ascending node" is an angle, measured at the center of the earth, from the vernal equinox to the ascending node.



At the diagram the reference point (vernal equinox) is marked by the (blue) arrow (see "X Achse") at the x-axis on the equatorial plane (see "Bezugsebene").

The angle (" $\Omega$ ") is measured from the ascending node (see "Knotenlinie") to the x-axis (see "X Achse"), that points to the vernal equinox.

**LongOfPericenter** = The argument of the perigee in degrees which defines the position of the ellipse axis compared to the equatorial plane. It moves the angle, in the plane of the orbit, between the line of the node (which belongs to the plane of the equator) and the large axis of the ellipse.

Alternately you can use this element:

**ArgOfPericenter** = Angle from "Node" to Pericenter (in degrees).

**ArgOfPericenter** is yet another fancy word for angle. Now that we've oriented the orbital plane in space, we need to orient the orbit ellipse in the orbital plane. We do this by specifying a single angle known as "argument of perigee".

A few words about elliptical orbits:



The point (P), where the satellite is closest to the earth, is called perigee, although it's sometimes called periapsis or perifocus.

The point (A), where the satellite is farthest from earth, is called apogee (aka apoapsis, or apifocus).

If we draw a line from perigee (P) to apogee (A), this line is called the line-of-apsides. (Apsides is, of course, the plural of apsis.) I know, this is getting complicated again. Sometimes the line-of-apsides is called the major-axis of the ellipse. It's just a line drawn through the ellipse the "long way" (see "Absidenlinie").

The line-of-apsides passes through the center of the earth. We've already identified another line passing through the center of the earth: the line of nodes.



The angle between these two lines is called the argument of perigee.

In the diagram this angle is marked with " $\omega$ ".

Where any two lines intersect, they form two supplementary angles, so to be specific, we say that argument of perigee is the angle (measured at the center of the earth) from the ascending node to perigee ( $\Omega$  and  $\omega$ ).

When argument of perigee = 0, the perigee occurs at the same place as the ascending node. That means that the satellite would be closest to earth just as it rises up over the equator.

When argument of perigee = 180 degrees, apogee would occur at the same place as the ascending node. That means that the satellite would be farthest from earth just as it rises up over the equator.

By convention, argument of perigee is an angle between 0 and 360 degrees.

**MeanLongitude** = The average anomaly corresponds to the position of a satellite in its orbit around the Earth compared to the axis of the perigee. Alternately you can use this element:

**MeanAnomaly** = The location of the body in the orbit at the specified Epoch.

Now that we have the size, shape, and orientation of the orbit firmly established, the only thing left to do is specify where exactly the satellite is on this orbit ellipse at some particular time.

**Epoch** (see below) specified a particular time, so all we need to do now is specify where, on the ellipse, our satellite was exactly at the Epoch time.

Anomaly is yet another astronomer-word for angle. Mean anomaly is simply an angle that marches uniformly in time from 0 to 360 degrees during one revolution. It is defined to be 0 degrees at perigee, and therefore is 180 degrees at apogee.



If you had a satellite in a circular orbit (therefore moving at constant speed) and you stood in the center of the earth and measured this angle from perigee, you would point directly at the satellite (view-line  $u$ ).

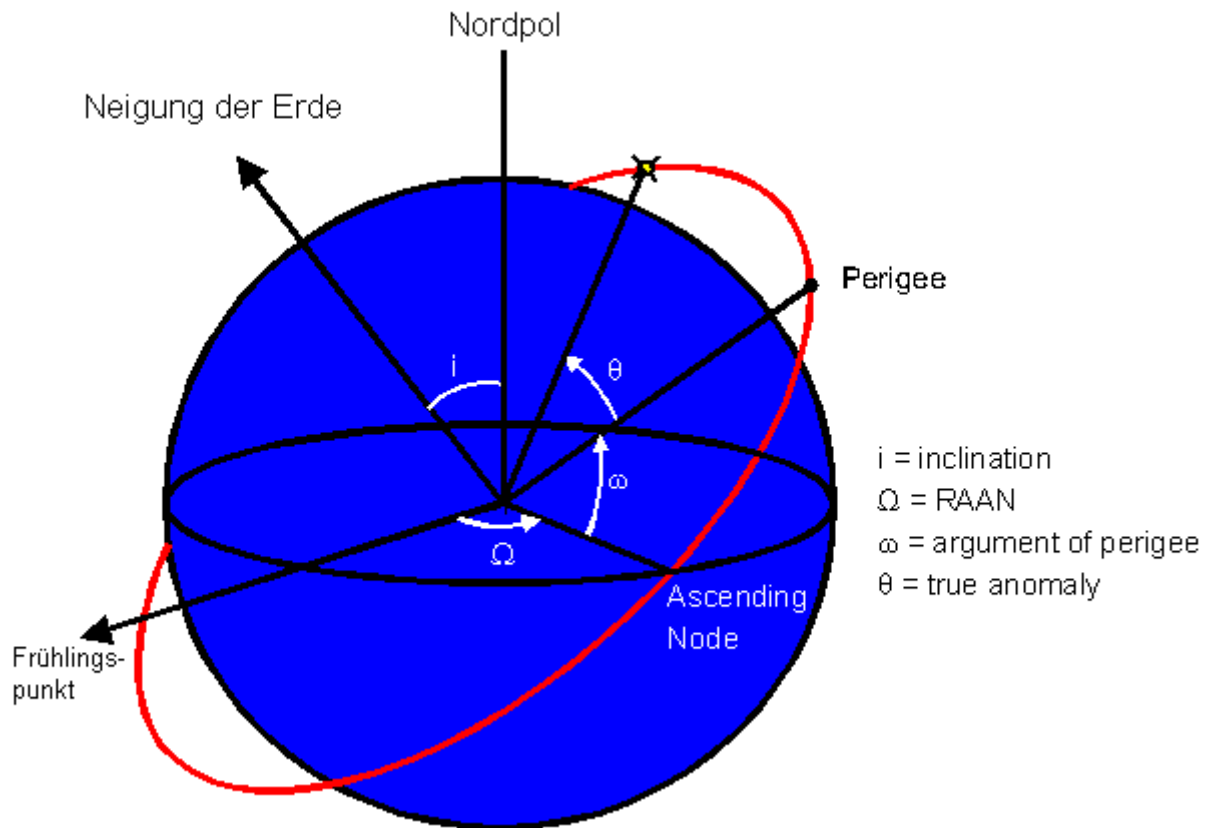
Satellites in non-circular orbits move at a non-constant speed, so this simple relation doesn't hold.

This relation does hold for two important points on the orbit, however, no matter what the eccentricity. Perigee always occurs at MA = 0, and apogee always occurs at MA = 180 degrees.

**Epoch** = A number which represents the desired time frame. This number is normally defined in terms of Julian Dates, but Universal Time may also be used. Please note that this element is optional and should NOT be used unless you wish the user's initial view of an object's orbit to ALWAYS be at the same time. It is defined in terms of days and decimal fractions of days. In the example above, the author is describing Venus' orbit as of January 1, 2000 at noon. Be aware that January 1st, 2000 is Celestia's (Version 1.3.x) current default epoch value.

A set of orbital elements is a snapshot, at a particular time, of the orbit of a satellite. Epoch is simply a number which specifies the time at which the snapshot was taken.

Finally some connections will be again clarified on the basis of a diagram:



Again a diagram of the German-speaking guide.

Nordpol = North pole

Neigung der Erde = Angular momentum of the earth (obliquity)

Frühlingspunkt = vernal equinox

## SemiAxes

**SemiAxes** (optional) = The default spherical shape of a Star can be modified by this optional declaration. Thus you could for example simulate the influence of another orbiting stellar body on a star.

SemiAxes specifies the relative sizes of a Star's radii in three dimensions:

Example 1: **SemiAxes** [1 1 0.5]

Example 2: **SemiAxes** [1 0.3 0.3]





Example 1: **SemiAxes** [1 1 0.5]

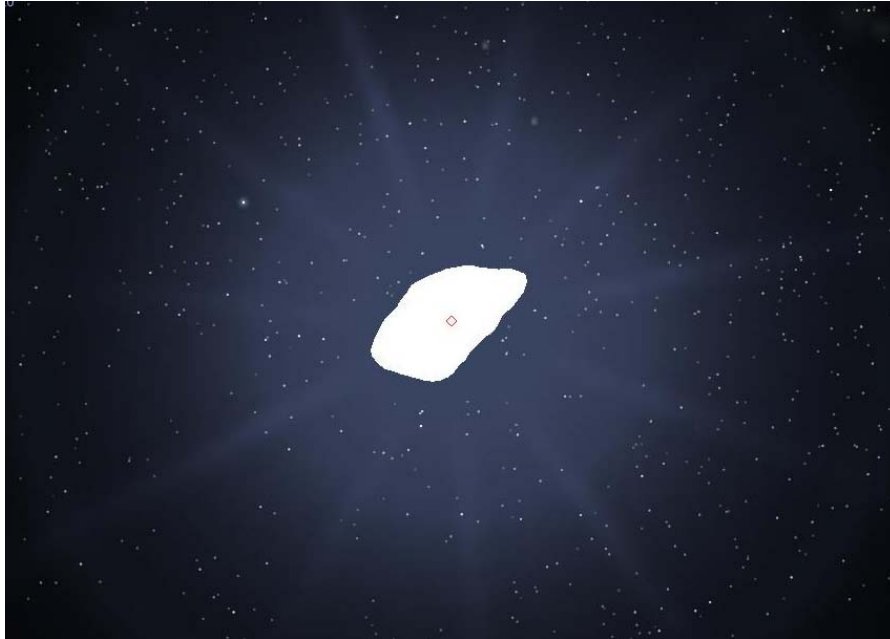


Example 2: **SemiAxes** [1 0.3 0.3] (front view left picture, side view right picture)

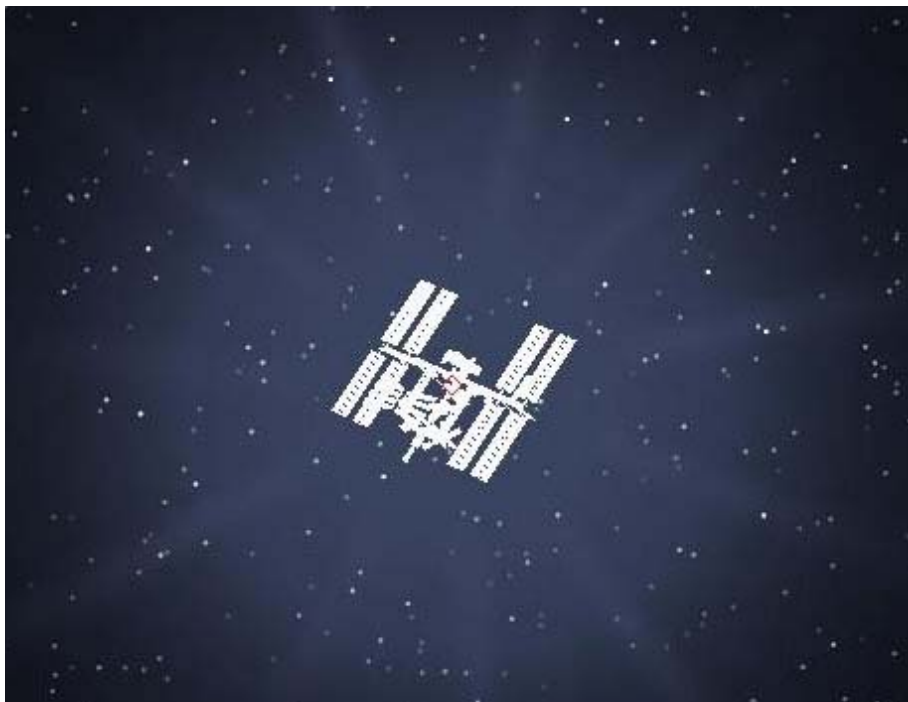
## Mesh

**Mesh** (optional) = The default spherical shape of a Star can be replaced by using the optional model declaration Mesh. So the star can also be deformed like in the examples above. But there are also more, let's say 'artistic' possibilities. Meshes can as usual be .3ds or .cmod-files.

Example: **Mesh** "MyMesh.3ds"



A sun with the mesh of Amalthea



The one and only ISS star!! In this case the mesh of the ISS has been used.

## Texture

**Texture** = The default surface texture of a Star (normally determined from its SpectralType) can be replaced by using the declaration Texture.

But care should be taken when using other textures with different star types, since only the texture will be changed and **not** also its spectral color. What happens when this is not taken into account shows the following picture of an O7V star with the texture of a brown dwarf. Not very convincing, but perhaps that will get changed in a later version of Celestia, I hope.





Example: **Texture** "browndwarf.jpg"

Be also advised that at the moment all textures for general star types have to be placed in the default Textures-folders of Celestia itself to be used.  
(i.e. "C:\programs\Celestia\textures\medres")

## Rotational Elements

The orientation of a Star is controlled by the same optional rotational element declarations as are used in SSC catalogs. These explanations have again been taken from the manual for .SSC-files by Ulrich Dickmann and Bob Hegwood and just slightly modified.

**RotationPeriod** (optional) = A number which defines the amount of time it takes for the selected object to fully rotate once around its axis. This value is defined in terms of hours.

Example: **RotationPeriod** 480

**Obliquity** (optional) = A number which defines the slant of the object's axis of rotation. The Obliquity parameter tilts the object through the specified number of degrees relative to the local reference plane. But this tilt is always made in the plane of the local coordinate 0, while suns and barycenters have their own axis, which can be tilted in any direction. Obliquity defines the amount of tilt, **not** the direction of the tilt. For this the element **EquatorAscendingNode** is needed.

This number can be defined with either a positive or a negative value.

Example: **Obliquity** **-12.043**

**EquatorAscendingNode** (optional) = Rotates the tilted object (defined in **Obliquity**) in the reference plane through the specified number of degrees. With the use of both elements (**Obliquity** and **EquatorAscendingNode**) the rotation axis of the object is now pointing in the right direction.

**RotationOffset** (optional) = Rotation of the object, related to the time period defined by the **Epoch** element.

Celestia creates objects with their main meridians aligned with the local coordinate 0 at the time defined by **Epoch**. This often leads to the situation, that the object points into the wrong direction, compared to reality.

**RotationOffset** defines through how many degrees the object has to be rotated (counterclockwise) to align it correct.

**PrecessionRate** (optional) = A number which defines the rate of precession of the axis of rotation in rad/days.

**Orientation** (optional) = Defines the orientation of an object within Celestias universal coordinate system.

The **Orientation** element contains the following syntax

Syntax: **Orientation** **[Angle x-axis y-axis z-axis]**

The angle can be defined from -360 to 360 degrees. The numbers for the axes (x, y, and z) define a vector, which represents the middle axis, around which the object is rotated. It is uncommon to use values greater than 1 or smaller as -1 for the axes. The rotation is clockwise. Negative values invert the direction of rotation to counterclockwise.

The following definitions are good for simple rotations around one axis:

**Orientation** **[Angle 1 0 0]**  
(rotates the object around the x-axis)

**Orientation** **[Angle 0 1 0]**  
(rotates the object around the y-axis)

**Orientation** **[Angle 0 0 1]**  
(rotates the object around the z-axis)

Other SSC parameters are not supported. In particular, Bumpmaps, Normalmaps, Nightlights and Atmosphere (including Clouds) are ignored.  
Also AltSurface and Location declarations are not (yet?) supported.

Now go on and have fun creating a whole universe with Celestia.

## **Credits**

Many thanks to Chris for implementing the new star features. And also many thanks to Fridger and the others for pushing the development, so that we finally have a version 1.4.0.

Special Thanks to Ulrich “Adirondack” Dickmann for giving me permission and especially the already written explanations from his and Bob Hegwoods SSC-Guide.

I also want to thank Selden for giving me details on the new features, which are supported and which not (yet?).