Astronomy Plugin

For Unreal Engine

**Description**

The astronomy plugin provides a blueprint interface to do many kids of astronomical calculations. For example, you can calculate star and planet positions from any point and time on earth or model the solar system. The plugin has over 30 advanced astronomical functions accessible through blueprints, and can use three kinds of planet ephemeris: JPL Low precision formulas, VSOP87, and high-precision NASA bsp files.

**Technical Overview**

The core of the plugin is a static blueprint library of over 30 astronomical functions. The plugin is designed to integrate with your blueprints and artwork, the star and planet blueprints that come with the plugin are low quality and intended for demonstration purposes. It is not recommended to use a SkyBox if you are placing star and planet actors – those objects will be in from of any skybox that your scene uses, so the stars and planets will appear in front of any clouds.

This plugin will only work with windows.

The Astronomy plugin uses a right hand coordinate system as most astronomy coordinate system are right handed. When needing to get an Unreal world coordinate, it is recommended to invert the Y axis. Some functions return this automatically as described in function documentation. The main functions that do not do this automatically are functions that directly return ephemeris data (like GetPlanetState).

**ExampleProject**

The plugin comes with an example project demonstrating a wide variety of the plugin’s features. Start the Example project by setting BP\_AstroGameModeExample as your projects startup game mode, and then open either the ‘Ground’ map for an earth surface perspective, or the ‘Space’ map for the solar system.

**Quick Start Guide**

This is how to quickly create sun, star, planets, and moon from an earth surface perspective. A youtube overview is available at <https://www.youtube.com/edit?o=U&video_id=OQPbqlKxTjM>

**Ephemeris**

The Astronomy plugin comes with three types of ephemeris.

JPLLowPrecision – These are low precision formulas provided by the Jet Propultion Laboratory. By default the plugin uses this ephemeris.

VSOP87 – An analytical series for planets and the sun.

JPLDE– SPK files are NASA generated ephemeris files. This will provide the most accurate and quickest ephemeris lookup, but will require additional setup (see setup SPK Ephemeris section).

**Changing Ephemeris**

To use a different ephemeris you need to call three functions below. There is an example in the BP\_AstroGameModeExample blueprint under the SetNewEphemeris function.

CreateNewEphemeris – This creates an ephemeris object of the desired ephemeris type.

InitEphemeris – This preforms any initialization required for the ephemeris. Optionally, if using SPK files, you can pass in an array of SPK files to load.

SetActiveEphemeris – This tells the plugin to use this ephemeris.

**SPK Setup**

1. Create a folder in your projects Content folder called “SPKData”
2. At a minimum, you need a leap second file (.tls) and a planet ephemeris file (.bsp).
   1. BSP Ephemeris files available for download at <ftp://ssd.jpl.nasa.gov/pub/eph/planets/bsp/>
      1. Download an ephemeris file and rename it “PlanetData.bsp”
   2. TLS (Leap Second time kernal) files available for download at <https://naif.jpl.nasa.gov/pub/naif/generic_kernels/lsk/>
      1. Download the latest tls file and rename it “TimeData.tls”

You can load additional satellite, comet, or other data by using the LoadKernal function of the JPLDE ephemeris object. The example project does this with Voyager 2 data.

After you build and package your project, you will need to copy the SPKData folder to the built content directory. You can have Unreal do this automatically in the build process by setting "Additional Non-Asset Directories to Copy" package setting.

**Important Blueprint Functions**

* Calculate Planet Direction From Earth - Returns the direction towards a planet, sun, or moon from a point on earth. Works by calling CalculatePlanetPosition to get a topocentric celestial position, then passes the result into TopocentricToAzEl().
* Get Planet State – Returns a planet/moon/sun positon from a solar system perspective. Do not use for an earth scene. Returns right handed coorcinate system.
* StarManager->UpdateAllStarPositions() – Updates all the stars associated with the star manager based on any updated UTCTime or Location changes made in the Star Manager properties.

**Advanced Star Options:**

* The *Star Class Table* defines which blueprint should be spawned for different star brightness. So you could add entry’s to this table if you wanted to spawn a different blueprint for stars that are dimmer or brighter.
* The *Star Size Table* defines a size transform based on the stars brightness. By default, brighter stars are made bigger to appear brighter.
* *SpawnDistancce* defines how far away to spawn the star blueprint objects.
* *Star Calculation Methods:*
  + Full does all calculations for each star. Use this method if stars do not need to be updated very often.
  + *Fast Update* calculates each star location once, then future updates calculate the earths rotation and moves the stars based on the earths rotation. Use this method if you need stars to be frequently updated.

**Acknowledgments:**

Special thanks to the US Naval Observatory for providing the NOVAS library and publishing the Astronomical Almanac.

Special thanks to the Jet Propulation Laboratory for providing planetary ephemeris.

Special thanks to NASA’s Navigation and Ancillary Data team for the SPICE toolkit.

Special thanks to <https://www.solarsystemscope.com/> for consolidating planet textures.

**Core function List:**

CalculateStarPosition - Returns the celestial position of a star (or any object outside the solar system) in various coordinate systems

CalculatePlanetPosition - Returns the celestial position of a planet in various coordinate systems

PlanetDirectionFromEarth - Returns the direction towards a planet from earth surface

GetEarthRotator - Returns a rotator (Left hand coordinate system to be compatible with Unreal) representing the earths rotation.

GetPlanetState - Returns a velocity and position vector of the selected planet, using the active ephemeris object.

**Other functions:**

GetNumberOfLeapSeconds – Number of leap seconds added to UTCTime

GetJulianDate – Gets a julian date from a calander date

GetCalendarDate – Gets calendar date from a julian date

GetTTJulianDatefromUTC – Gets terrestrial time (aka ephemeris time) in julian date format

GetUT1JulianDatefromUTC - Returns the approximate (within 0.5 seconds) UT1 time in Julian Date format

GetCoordsTypeName - Returns the name of the type of coordinate returned by Star or Planet position, based on observer location type and coord system selections. (examples are Topocentric place, astronomic place, etc)

CalculateSideRealTime – Calculates sidereal time.

TerrestrialToCelestial - rotates a vector from the terrestrial (ITRS) to the celestial (GCRS) system.

CelestialToTerrestrial - rotates a vector from the celestial (GCRS) system to the terrestrial (ITRS)

TopocentricToHorizon - Transforms topocentric right ascension and declination to zenith distance and azimuth. (I.e. gets the azimuth and elevation from a celestial position)

AzElToRectangular - Transforms Azimuth and elevation to cartesian rectangular coordinates (left hand rule for Unreal compatibility)

RectangularToAzEl- Transforms cartesian rectangular coordinates to Azimuth and elevation

TransformCoordSystem - Transforms an object for a change of epoch and/or equator and equinox.

HipparcosToJ2000 - convert Hipparcos catalog data at epoch J1991.25 to epoch J2000.0.

PrecessionShift - Precesses equatorial rectangular coordinates from one epoch to another.

ConvertSphericalToEclipticLatLong - Convert right ascension and declination to ecliptic longitude and latitude.

EquationOfOrigins - Computes the true right ascension of the celestial intermediate origin (CIO) at a given TT Julian date.

EarthRotationAngle - This function returns the value of the Earth Rotation Angle (theta) for a given UTC Julian date.

SetCelestialPoleOffsets - This function specifies celestial pole offsets for very high - precision applications.

GetEarthTiltParameters - computes various quantities related to the orientation of the Earth’s rotation axis

ConvertToEcliptical - Converts an equatorial position vector to an ecliptic position vector

ConvertToEquatorial - Converts an ecliptic position vector to an equatorial position vector.

SphericalCoordsToVector - Converts a Right Ascenstion and a declination to a vector. (equatorial spherical coordinates to a vector (equatorial rectangular coordinates)

GetPlanetToEarthRadiusRatio - Returns the ratio of a planet's radius compared to earth.

GetKMFromAU - Returns a FDoubleVector that is in AU units in KM

GetAzElInDegrees - Returns the azimuth (0-360 degrees) and elevation from a world direction vector.