SunLight – Location based Time of Day

Description:

Calculates the light color and orientation of a light source based on geographical location and time of day to create a realistic sun light source. Works with the Unity's dynamic skybox introduced in Unity 5. Simple to use; just drag it into your scene and let SunLight handle the directional light.

SunLight creates two directional lights, one to illuminate the scene and another one is used to trigger Unity's Skybox only (layer mask set to nothing). This is necessary to avoid unwanted directional illumination from below the horizon in twilight conditions.

Supports C# and Unity Script (JS)

New in version 1.2: Removed refraction when sun is close to horizon to avoid a (one frame) visual issue.
Unity 5.5. and above: Uses RenderSettings.sun to set the sun in light settings.
New in version 1.1: SunLight.northDirection : float – to set a custom simulated north direction in degrees.

Usage:

Drag the prefab (C# and JS based prefabs available) into the scene and set location and time in inspector.

– Chose a location from 10 presets or manually set latitude and longitude.
– Choose whether to let SunLight handle the progress of time.
– If you wish you can manually override the azimuth (horizontal rotation) of the sun via inspector or in your own script.

Please note:
Prior Unity 5.5 Unity does not offer a way to set the directional light from the skybox by scripting yet - if not set manually it will try to use the brightest directional light in the scene automatically.
SunLight sets the intensity of its SkyboxLight (which does not illuminate any objects) to 8.0 – please keep the intensity of all other directional lights in your scene below or at 0.5 otherwise Unity's skybox might be triggered by the wrong light.

Script Reference:

Namespace: Hessburg (C# only)

SunLight.latitude : float

Sets latitude (valid range: -90.0 to 90.0)

SunLight.latitude : float

Sets longitude (valid range:-180.0 to 180.0)

SunLight.offsetUTC : float

Sets the timezone by offset from UTC (valid range: -12.0 to 14.0)

SunLight.dayOfYear : int

Sets the day of the year (valid range: 1 to 365 or 1 to 366 for leap years)

SunLight.timeInHours : float

Sets time in decimal hours (valid range: 0.0 to 24.0)
Example: 8.5 in decimal hours equals 12:30 PM

SunLight.SetTime(int, int, int)

Sets time in human-readable format (Hours : int, Minutes : int, Seconds : int)

SunLight.progressTime : boolean

Toggles the progress of time and date automatically. SunLight.progressTime to true to enable.

SunLight.timeProgressFactor : float

Controls how fast time will progress. For real time, set this value to 1.0 (valid range => 0.0)
Ignored if SunLight.progressTime is set to false.

SunLight.northDirection : float

Sets a custom simulated north direction in degrees. (valid range: 0.0 to 360.0, initial value = 0.0)

SunLight.overrideAzimuth : boolean

Controls whether the calculated horizontal rotation of the light is overridden by SunLight.artisticSunAzimuth. Set to true to override.

SunLight.artisticSunAzimuth : float

Horizontal rotation of the light in degrees.(valid range: 0.0 to 360.0)
Ignored if SunLight.overrideAzumuth is set to false.

SunLight.GetSceneLight() : Light

Read only – returns the scene light
(A directional light instantiated by SunLight to illuminate the scene which never goes below the horizon to avoid lighting from below)

SunLight.GetSkyboxLight() : Light

Read only – returns the Skybox light
(A directional light instantiated by SunLight to trigger the Unity skybox which layer mask is set to nothing)

SunLight.GetLensFlare() : LensFlare

Read only – returns the Lens Flare
(A LensFlare instantiated by SunLight to simulate the sun)

SunLight.GetSunAzimuth() : float

Read only – returns the azimuth (horizontal rotation) of the sun (SkyboxLight)
0.0 equals north, 90.0 equals east, 180.0 equals south, 270.0 equals west

---

SunLight.GetSunAltitude() : float

Read only – returns the altitude of the sun (SkyboxLight)
0.0 is horizon 90.0 is zenith

SunLight.leapYear : boolean

Controls whether the simulated year is a leap year. Set to true for leap years.

Contents:

– “Prefabs“ folder contains the C# and JS prefabs – drag one of them into in your scene
– „Demo“ folder contains a simple demo scene, its demo scripts and materials
– “Resources“ folder contains the prefabs for the lights and flares. It also contains a JSON file with precalculated values for SunLight
– “Scripts“ folder contains the source code of C# and JS variants of SunLight
– “Flare“ folder contains a lens flare and its texture.
– “Experimental“ Folder – content of this folder is not advertised, no support will be given, use as is! See below:

Please note:

SunLight increases the _AtmosphereThickness parameter of the Skybox Shader during simulated night times.
When you disable SunLight the thickness settings won’t reset. This could lead to a reddish skybox at simulated daytime.
Just set back the thickness parameter of the Skybox Shadert in standard settings in your own script then:

```
UnityEngine.RenderSettings.skybox.SetFloat("_AtmosphereThickness", 1.0);
```

Experimental: (use as is – no support)

Overcast – a modified version of the original Unity Procedural Skybox Shader
It simulates an overcast sky (in a very primitive way) by converting the color of the sky to grayscale.
The amount of grayscale depends on the float *“_HessburgOvercast“* of the SkyboxOvercast shader.

Namespace: Hessburg (C# only)

SunLight.useOvercastSkyboxShader : boolean

Experimental:
Controls whether the sky uses the modified variant of the Unity 5 procedural skybox shader. Set to true to activate.

SunLight.overcastFactor : float

Experimental: Simulates the effect of overcast sky on the sun light (valid range 0.0 – 1.0)
Additionally it controls the desaturation of the color of the sky if SunLight.useOvercastSkyboxShader is set to true.

Support:

assets.support @ hessburg.com