AzureCoreSystem:

class in UnityEngine.AzureSky / Inherits from:MonoBehaviour

Description:

This class is the main component that handles the entire system.

Properties:

timeSystem	Instance of the ***AzureTimeSystem*** class used to handle the time stuff in this component.
weatherSystem	Instance of the ***AzureWeatherSystem*** class used to handle the weather stuff in this component.
eventSystem	Instance of the event system class used to handle the time and date events.
coreUpdateMode	The way the Core System should be updated.
timeSinceLastUpdate	The time passed since the last core system update. (Read Only)
refreshRate	The time interval used to update the core system.
onCoreUpdateEvent	The event that is invoked every time the core system is updated.
reflectionProbe	The reference to the reflection probe handled by this core system component.
followTargetList	List of the follow targets.

Methods:

UpdateReflectionProbe()	Update the reflection probe attached in the options tab. **Note:** It can be very slow if called every frame!!!
UpdateDynamicGI()	Update the environment cubemap texture from Unity’s DynamicGI. **Note:** It can be very slow if called every frame!!!

AzureTimeSystem:

class in UnityEngine.AzureSky / Inherits from:

Description:

Class that handles the time, date and location stuff. It is used as an internally instance of the AzureCoreSystem class.

Properties:

sunTransform	The transform that will represent the position of the sun in the sky.
sunElevation	Stores the sun elevation in the sky in the Range(-1.0, 1.0). (Read Only)
moonTransform	The transform that will represent the position of the moon in the sky.
moonElevation	Stores the moon elevation in the sky in the Range(-1.0, 1.0). (Read Only)
starfieldTransform	The transform that will represent the position of the starfield in the sky.
directionalLight	The directional light that will apply the sun and moon lighting to the scene.

timeMode	The time mode used to perform position of the sun and moon in the sky according to the time of day.
timeDirection	The direction in which the time of day will flow.
timeLoop	The time repeat mode cycle.
timeline	The timeline that represents the day cycle. Do not confuse with the time of day, they may not be the same thing depending on the configuration.
latitude	The north-south angle of a position on the Earth's surface.
longitude	The east-west angle of a position on the Earth's surface.
utc	Prior to 1972, this time was called Greenwich Mean Time (GMT). But is now referred to as Coordinated Universal Time or Universal Time Coordinated (UTC).
day	Represents the day in the calendar.
month	Represents the month in the calendar.
year	Represents the year in the calendar.
startTime	The value the timeline should start the scene when entering the play mode.
dayLength	The duration of the day cycle in minutes.
minLightAltitude	The minimmun directional light angle according to the horizon line. Useful for saving performance by avoiding stretched shadows when the light is horizontally tilted relative to the scene.
dawnTime	The time that marks the transition from nighttime to sunrise in the timeline cycle.
duskTime	The time that marks the transition from sunset to nighttime in the timeline cycle.
timeOfDay	The current time of day inside the timeline cycle. (Read Only)
hour	The hour according to the time of day. (Read Only)
minute	The minute according to the time of day. (Read Only)
evaluationTime	The time used to evaluate the curves and gradients.
timeLengthCurve	The curve used to evaluate the daytime and nightime length.
celestialBodiesList	The list of the celistial bodies the system can simulate other than the sun and moon.
weekNameList	String array that stores the name of each day in the week. (Read Only)
monthNameList	String array that stores the name of each month. (Read Only)
DayNumberList	Array with 42 numeric strings used to fill a calendar. (Read Only)
selectedCalendarDay	The current selected day in the calendar. (Read Only)
dayOfWeek	The number that represents the current selected day in the week (0 – 6). (Read Only)
daysInMonth	Stores the total number of days in the current month. (Read Only)
isPlayingTimelineTransition	Is a timeline transition current running? (Read Only)

Methods:

UpdateCelestialBodies()	Updates the transforms of the sun, moon and directional light according to the time of day.
IncreaseDay()	Increases a day in the calendar.
DecreaseDay()	Decreases a day in the calendar.
IncreaseMonth()	Increases a month in the calendar.
DecreaseMonth()	Decreases a month in the calendar.
IncreaseYear()	Increases a year in the calendar.
DecreaseYear()	Decreases a year in the calendar.
SetDate()	The proper way for setting a custom date.
GetTimeOfDayVector()	Returns the current time of day as a Vector2Int(hours, minutes).
GetTimeOfDayString()	Returns the current time of day as a string ("00:00").
GetDateVector()	Returns the current date as a Vector3Int(year, month, day).
GetDateString()	Returns the current date converted to string using the default date format used by Azure. Example "January 01, 2024".
GetDateString(string format)	Returns the current date converted to string using a custom format as parameter. Example Format: "MMMM dd, yyyy"
GetDayOfWeekString()	Returns the current day of the week as string.
GetDayOfWeekInteger()	Returns the current day of the week as an integer number between 0 and 6.
UpdateCalendar()	Used to adjust the calendar when there is a change in the date.
StartTimelineTransition()	Starts a timeline transition to a desired time.
UpdateTimeLengthCurve()	Used to Update the time length curve when there is a change in the dawn and dusk time.

AzureWeatherSystem:
class in UnityEngine.AzureSky / Inherits from:

Description:
Class that handles the weather features. It is used as an internally instance of the AzureCoreSystem class.

Properties:

weatherPropertyGroupList	The list containing the weather property groups.
globalWeatherList	The list containing the global weather presets.
weatherZoneList	List of local weather zones arranged according to its priorities.
thunderSettingsList	The list containing the settings for instantiation of the thunder prefabs.
currentWeatherPreset	The current weather preset.
targetWeatherPreset	Stores the target weather preset when runing a global weather transition. (Read Only)

weatherZoneTrigger	The trigger used to detect if it is entering a local weather zone.
evaluationTime	The time used to evaluate the curves and gradients. It is handled by the AzureCoreSystem by default.
isWeatherChanging	There is a global weather transition in progress? (Read Only)
globalWeatherIndex	Stores the current global weather index in use. (Read Only)
weatherTransitionProgress	Stores the progress of a global weather transition. (Read Only)

Methods:

UpdateWeatherSystem()	Performs a complete update in the weather system. Handled by the Update() of the AzureCoreSystem script.
OverrideTargetProperties()	Gets the output values from the weather properties and uses them to override the targets.
InstantiateThunderPrefab()	Instantiates a thunder prefab in the scene using as position the especification from the thunder settings list. When the thunder sound is over, the instance is automatically deleted.
SetGlobalWeather(int index)	Starts a global weather transition using the preset from the global weather list. It changes the weather index to the selected one.
WeatherGroupsEnableAll()	Enable all the weather groups.
WeatherGroupsDisableAll()	Disable all the weather groups.
WeatherGroupsEnableAt()	Enable a weather group at a given index.
WeatherGroupsDisableA()	Disable a weather group at a given index.
WeatherGroupsSetStateAt()	Set the active state of a weather group giving its index and state.
GetFloatOutput()	Returns the float output of a custom weather property.
GetColorOutput()	Returns the color output of a custom weather property.
GetVector3Output()	Returns the Vector3 output of a custom weather property.
InitializePropertyTargets()	PropertInfo and FieldInfo are not serializable, so the targets must always be reassigned when the scene starts. It is handled by the AzureCoreSystem.

AzureSkyRenderer:

class in UnityEngine.AzureSky / Inherits from:MonoBehaviour

Description:

This class handles the attributes used to render the sky and fog scattering effect.

Properties:

sunTransform	The transform used to represent the position of the sun in the sky.
moonTransform	The transform used to represent the position of the moon in the sky.
starfieldTransform	The transform used to represent the position of the starfield in the sky.
skyMaterial	The material that renders the sky.

fogMaterial	The material that renders the fog scattering effect.
sunTexture	The cubemap texture used to render the sun sphere.
moonTexture	The cubemap texture used to render the moon sphere.
starfieldTexture	The cubemap texture used to render the regular stars and the Milky Way.
constellationTexture	The cubemap texture used to render the sky constellation.
dynamicCloudTexture	The 2D texture used to render the dynamic clouds.
emptySkyShader	The shader used to render only the sky without clouds.
dynamicCloudShader	The shader used to render sky with dynamic clouds.
wavelength	The Vector3 that represents the wavelength of the visible light.
molecularDensity	The molecular density of the air.
kr	The rayleigh altitude in meters.
km	The mie altitude in meters.
rayleigh	The rayleigh scattering multiplier.
mie	The mie scattering multiplier.
mieDirectionalityFactor	The mie directionality factor.
scattering	The scattering intensity multiplier.
skyLuminance	The luminance of the sky when there is no sun or moon in the sky.
exposure	The exposure of the internal sky shader tonemapping.
rayleighColor	The rayleigh color multiplier.
mieColor	The mie color multiplier.
sunSize	The size of the Sun sphere in the sky.
sunOpacity	The opacity of the Sun sphere.
sunColor	The color multiplier of the Sun sphere.
moonSize	The size of the Moon sphere in the sky.
moonOpacity	The opacity of the Moon sphere.
moonColor	The color multiplier of the Moon sphere.
moonRotationOffset	The rotation offset to adjust the moon cubemap texture in its sphere.
starsIntensity	The intensity of the regular stars.
milkyWayIntensity	The intensity of the Milky Way.
starfieldColor	The color multiplier of the entire starfield.
skyExtinction	The extinction of the light coming from the outer space, caused by the atmosphere density.
constellationIntensity	The intensity of the stars constellation.
constellationColor	The color of the sky constellation.
starfieldRotationOffset	The rotation offset to adjust the starfield cubemap texture in the sky sphere.
mieDistance	The distance of the mie bright influence in the fog scattering effect.

globalFogDistance	The distance of the global fog scattering effect.
globalFogSmoothStep	The smooth step transition from where there is no global fog in the scene to where is completely foggy.
globalFogDensity	The density of the global fog scattering effect.
heightFogDistance	The distance of the height fog scattering effect.
heightFogSmoothStep	The smooth step transition from where there is no height fog in the scene to where is completely foggy.
heightFogDensity	The density of the height fog scattering effect.
heightFogStartAltitude	The height altitude where the height fog scattering effect should start.
heightFogEndAltitude	The height altitude where the height fog scattering effect should end.
fogBluishDistance	The distance of the bluish color effect of the fog at distance.
fogBluishIntensity	The intensity of the bluish color effect of the fog at distance.
heightFogScatteringMultiplier	The scattering multiplier based on the height fog.
dynamicCloudAltitude	The altitude of the dynamic clouds in the sky.
dynamicCloudDirection	The movement direction of the dynamic clouds.
dynamicCloudSpeed	The movement speed of the dynamic clouds.
dynamicCloudDensity	The first color of the dynamic clouds.
dynamicCloudColor2	The second color of the dynamic clouds.
dynamicCloudColor2	The second color of the dynamic clouds.
dynamicCloudUV	The dynamic cloud uv. (Read Only)
updateMode	How the shader uniforms of the sky and fog material should be updated? Locally every frame or from an external call?
cloudMode	How the clouds should be rendered.

Methods:

InitializeSkySystem()	Sets the shader uniforms that only requires one update at start.
UpdateSkySystem()	Sets the shader uniforms that requires constantly update.

AzureNotificationCenter:
class in UnityEngine.AzureSky / Inherits from:

TimeSystemDelegate(AzureTimeSystem timeSystem);

OnTimelineChanged	Event triggered every time the timeline changes in the AzureCoreSystem's time system.
OnMinuteChanged	Event triggered every time the minute changes. *(in game time)*
OnHourChanged	Event triggered every time the hour changes. *(in game time)*
OnDayChanged	Event triggered every time the day changes. *(in game date)*
OnMonthChanged	Event triggered every time the month changes. *(in game date)*
OnYearChanged	Event triggered every time the year changes. *(in game date)*

WeatherSystemDelegate(AzureWeatherSystem azureWeatherSystem);

OnBeforeOverrideUpdate	Event triggered just before the weather system overrides the target properties.
OnAfterOverrideUpdate	Event triggered just after the weather system overrides the target properties.
OnBeforeWeatherSystemUpdate	Event triggered just before the weather system is updated.
OnAfterWeatherSystemUpdate	Event triggered just after the weather system is updated.
OnWeatherTransitionEnd	Event triggered just after the weather system ends a global weather transition.

VolumetriLightPreRenderDelegate(AzureVolumetricLightRenderer renderer);

OnVolumetricLightPreRender	Event used by the volumetric light effect system to link the light draw mesh function to the command buffer that renders the light meshes in the camera's 'BeforForwardAlpha' event. The command buffer must be filled in the OnPreRender event of the camera rendering the volumetric lights, but the OnPreRender event works only in the scripts attached to the camera. So we need to use this custom event as a link from the volumetric light script to the camera OnPreRender event.

```
using UnityEngine;
using UnityEngine.AzureSky; // Always include it to be able to access the Azure[Sky] classes.

public class Example : MonoBehaviour
{
    // Registering to the OnDayChanged event.
    private void OnEnable()
    {
        AzureNotificationCenter.OnDayChanged += OnDayChanged;
    }

    // Unregistering from the OnDayChanged event.
    private void OnDisable()
    {
        AzureNotificationCenter.OnDayChanged -= OnDayChanged;
    }

    // This method will be executed when the date changes in the time system!
    private void OnDayChanged(AzureTimeSystem azureTimeSystem)
    {
        Debug.Log(azureTimeSystem.GetDateString());
    }
}
```

How to Change the Global Weather by Scripting:

```
using UnityEngine;
using UnityEngine.AzureSky; // Always include it to be able to access the Azure[Sky] classes.

public class Example : MonoBehaviour
{
    // The reference to the core system component.
    // You need to reference it manually in the Inspector.
    public AzureCoreSystem azureCoreSystem;

    private void Start()
    {
        // Changing the global weather to the 'Storm' weather preset.
        // Note that in the 'Global Weather List', by default the 'Storm' weather preset...
        // is attached to the element number 7. We need to pass as parameter, the element number
        // of the target weather preset in the global weather list.
        azureCoreSystem.weatherSystem.SetGlobalWeather(7);
    }
}
```

How to Change the Timeline by Scripting:

```
using UnityEngine;
using UnityEngine.AzureSky; // Always include it to be able to access the Azure[Sky] classes.

public class Example : MonoBehaviour
{
    // The reference to the core system component.
    // You need to reference it manually in the Inspector.
    public AzureCoreSystem azureCoreSystem;

    private void Update()
    {
        if (Input.GetKeyUp(KeyCode.Space))
        {
            // Accessing the time system instance inside the core system and setting the
            // timeline to 12 hours, note that the timeline requires a value between 0 and 24.
            azureCoreSystem.timeSystem.timeline = 12.0f;
        }
    }
}
```

How to Change the Date by Scripting:

```
using UnityEngine;
using UnityEngine.AzureSky; // Always include it to be able to access the Azure[Sky] classes.

public class Example : MonoBehaviour
{
    // The reference to the core system component.
    // You need to reference it manually in the Inspector.
    public AzureCoreSystem azureCoreSystem;

    private void Update()
    {
        if (Input.GetKeyUp(KeyCode.Space))
        {
            // Accessing the time system instance inside the core system and setting the
            // date to January 1, 2024. Note the parameters ordering (day, month, year).
            azureCoreSystem.timeSystem.SetDate(1, 1, 2024);
        }
    }
}
```