Time of Day

Generated by Doxygen 1.8.6

Sat Jun 7 2014 16:32:18

Contents

Chapter 1

Main Page

1.1 About

Time of Day is a package to render realistic dynamic sky domes with day and night cycle, clouds, cloud shadows, weather types and physically based atmospheric scattering.

Sky:

- Physically based sky shading
- Rayleigh & Mie scattering
- Highly customizable
- Dynamic weather manager
- Presets for various locations & planets
- Specially optimized sun shafts

Clouds:

- Dynamic wind speed & direction
- Configurable shape, color & scale
- Correctly projected cloud shadows

Time & Location:

- Dynamic day & night cycle
- Full longitude, latitude & time zone support
- Full Gregorian calendar support
- Continuously animated moon phases

Performance & Requirements:

- Extremely optimized shaders & scripts
- Supports shader model 2.0
- Supports all platforms

- Supports linear & gamma color space
- Supports forward & deferred rendering
- Supports HDR & LDR rendering
- Supports virtual reality hardware

[Forum Thread | Web Player | Documentation]

You can expect a thoroughly documented, well-written and highly optimized code base.

All equations used in the shaders and scripts include references to the scientific papers they are based on.

1.2 Getting Started

1. Add the sky dome to your scene:
   - Drag the prefab found in "Sky Assets/Prefabs" into your scene
   - If your scene uses fog, it is recommended to enable "World -> Set Fog Color" in the prefab inspector
   - If your scene uses ambient light, it is recommended to enable "World -> Set Ambient Light" in the prefab inspector
   - Tweak the other parameters until you are satisfied with the result
2. Move the sky dome to the camera position in every frame:
   - Select your camera and add the Time of Day camera script (Component -> Time of Day -> Camera Main Script)
   - Drag your instance of the sky dome onto the the "Sky" property of the script
   - Set "Dome Pos To Camera" to enabled if it is not already
   - Tag this camera game object as "MainCamera"
3. Automatically fit the sky dome size to the far clip plane:
   - Go to the Time of Day camera script of your camera
   - Set "Dome Scale To Far Clip" to enabled
   - Set "Dome Scale Factor" to a value suitable for your specific scene and platform
4. Render sun shafts on the main camera:
   - Select your camera and add the Time of Day sun shaft script (Component -> Time of Day -> Camera Sun Shafts)
   - Drag your instance of the sky dome onto the the "Sky" property of the script
   - Tweak the parameters until you are satisfied with the result

REMARK: The camera script moves the sky dome directly before clipping the scene, guaranteeing that all other position updates have been processed. You should not move the sky dome in "LateUpdate" because this can cause minor differences in the sky dome position between frames when moving the camera.

1.3 Time & Location Setup

The sky dome prefab has a script TOD_Time attached to it that manages the dynamic day & night cycle. Enabling and disabling this script enables and disables the automatic cycle.

The following parameters are being set by the script:

- TOD_Sky.Cycle.Hour

- TOD_Sky.Cycle.Day
- TOD_Sky.Cycle.Month
- TOD_Sky.Cycle.Year
- TOD_Sky.Cycle.MoonPhase

1.4 Weather Manager

The script TOD_Weather can be used to automatically set various parameters of TOD_Sky according to weather presets.

The following parameters are being set by the script:

- TOD_Sky.Atmosphere.Fogginess
- TOD_Sky.Clouds.Density
- TOD_Sky.Clouds.Sharpness
- TOD_Sky.Clouds.Brightness

1.5 Time & Location Setup

The TOD_Sky.Cycle parameter section allows for configuration of the sky dome to represent the exact sun movement and day length of any location on the planet depending on Gregorian date, UTC/GMT time zone and geographic coordinates. It is important to set the correct time zone with the longitude and latitude of the location to guarantee consistent results with the real world. All of those parameters are completely optional though - if the sky dome should be used in a generic fantasy world they can simply be ignored and left at their default values.

1.6 Quality Levels

There are various different quality levels for both the sky dome and the cloud shader. Those quality settings can be configured both dynamically at runtime and directly in the Unity editor window using two inspector enums.

TOD_Sky.CloudQuality:

- Bumped offers complex cloud shading with dynamic density and cloud normal mapping
- Density offers simplified cloud shading with dynamic density but without normal mapping
- Fastest offers extremely simplified cloud shading with simplified cloud shape calculations

TOD_Sky.MeshQuality:

- High sky dome and moon mesh vertex count
- Medium sky dome and moon mesh vertex count
- Low sky dome and moon mesh vertex count

1.7 Performance Remarks

- The size of a web player with just the sky dome is only around 200KB as most equations are evaluated dynamically
- All scripts and shaders are highly optimized and will not cause significant FPS drops on desktop computers
- Older mobile devices should switch to the cloud and dome quality settings that offer suitable performance
- The cloud shadows utilize a Unity projector and require another draw call for all objects they are projected on
- The sun shaft image effect is the only component that requires Unity Pro, everything else works just as well on Indie
- Reducing the texture resolution in the cloud texture import settings can increase performance on mobile

1.8 Rendering Order

All components of the sky dome are being rendered after the opaque but before the transparent meshes of your scene.That means that only areas of the sky dome that are not being occluded by any other geometry have to be rendered.

The rendering order of the sky dome components is the following:

- Transparent-500 Unity Skybox (if your scene uses one)
- Transparent-490 Space Dome (only at night)
- Transparent-480 Moon Mesh (only at night and if visible)
- Transparent-470 Atmosphere (if not manually disabled)
- Transparent-460 Sun Plane (only at day and if visible)
- Transparent-455 Clear Alpha (if sun shafts are enabled)
- Transparent-450 Cloud Dome (if not manually disabled)

This leads to 2-5 draw calls to render the complete sky dome depending on the scene setup.

1.9 Global Shader Parameters

There are some global shader parameters set by the TOD_Sky script that can be used in your custom shaders.

Gamma values:

- TOD_Gamma = TOD_Sky.Gamma
- TOD_OneOverGamma = TOD_Sky.OneOverGamma

Various colors:

- TOD_SunColor = TOD_Sky.SunColor
- TOD_MoonColor = TOD_Sky.MoonColor
- TOD_LightColor = TOD_Sky.LightColor
- TOD_CloudColor = TOD_Sky.CloudColor
- TOD_AdditiveColor = TOD_Sky.AdditiveColor

- TOD_MoonHaloColor = TOD_Sky.MoonHaloColor

World space direction vectors:

- TOD_SunDirection = TOD_Sky.SunDirection
- TOD_MoonDirection = TOD_Sky.MoonDirection
- TOD_LightDirection = TOD_Sky.LightDirection

Sky dome object space direction vectors:

- TOD_LocalSunDirection = InverseTransformDirection(TOD_Sky.SunDirection)
- TOD_LocalMoonDirection = InverseTransformDirection(TOD_Sky.MoonDirection)
- TOD_LocalLightDirection = InverseTransformDirection(TOD_Sky.LightDirection)

Those variables can be used in your custom shaders by simply defining uniform variables with the same name. Those will then automatically be filled with the correct values. The Time of Day shaders call pow(color, TOD_OneOverGamma) on the result colors of either vertex or fragment shaders to assure consistent visuals in both linear and gamma space.

1.10 Presets

There are several presets that can be applied by clicking the "Choose Preset" button in the TOD_Sky inspector.

1.11 Examples

The package comes with various example scripts to demonstrate sky dome interaction.

- AudioAtDay / AudioAtNight / AudioAtWeather: Fade audio sources in and out according to daytime or a specific weather type
- ParticleAtDay / ParticleAtNight / ParticleAtWeather: Fade particle systems in and out according to daytime or a specific weather type
- RenderAtDay / RenderAtNight / RenderAtWeather: Enable or disable renderer components according to daytime or a specific weather type
- DeviceTime: Automatically set the daytime to the device time on scene start
- SkyboxGenerator: Renders the world around a camera to 6 static images to use in cubemaps or as a Unity skybox

1.12 Frequently Asked Questions

Q: How can I use the sky dome with the virtual reality devices like the Oculus Rift?

- Add the TOD_Camera script to one of the cameras (preferably the one that's being rendered first)
- The sky will render correctly without duplicate images or artifacts

Q: How can I use custom skybox at night?

- Disable the child object called "Space"

- Setup your camera to render the skybox of your choice, it will automatically be visible at night

Q: How can I align the cardinal directions with those of the scene?

- Rotate the sky dome around the y-axis such that the sun rises in the east of your scene
- Do not set the scale to negative values as this will lead to rendering artifacts

Q: How can I use lightmapping?

- Disable the shadows of the sun and moon directional lights
- Add a dummy directional light to bake your lightmaps from

REMARK: Lightmaps are not meant to be used with moving light sources as they always represent static lighting conditions. You should therefore always use a static dummy directional light to bake your lightmaps with. Alternatively, you can also set TOD_Sky.Light.MinimumHeight to 1 to force the sky dome light sources to always be at zenith.

Q: How can I synchronize the sky dome across multiple clients?

- To synchronize the cloud movement, synchronize the property TOD_Sky.Components.Animation.CloudUV of type Vector4
- To synchronize the cycle settings, synchronize the property TOD_Sky.Cycle.Ticks of tyle long

Q: How can I fix Z-fighting and sorting issues with the cloud shadows?

- Adjust the values for "Offset" directly in the shader code of the cloud shadow shaders

REMARK: Offset values have to be constants and can therefore only be adjusted directly in the shader code. Suitable values depend on the depth buffer resolution of the targeted platform and hardware. While the default values work in most scenarios, some scenes might require some further tweaking. If you are having issues with setting those values up correctly, please feel free to contact me.

Q: How can I fix Unity tree creator trees causing issues if cloud shadows are enabled?

- This is a bug of the tree creator shaders if placed using the terrain tools
- To fix it you have to use different shaders for your trees, like Nature/Tree Soft Occlusion Bark & Leaves

1.13 Contact Information

If you have any questions that cannot be answered using the FAQ or documentation feel free to contact me:

- In the official `forum thread` of the package
- Via `personal message` on the Unity community forums
- Via `Twitter`
- Via `my website`

REMARK: I should always be able to reply within 48 hours. If I did not reply after several days, please try using a different method to contact me as there might be an issue with the one you chose. If I am not available for multiple days I will always try to announce this beforehand in the offical forum thread.

1.14 Literature

The following literature has been used to implement physically correct atmospheric scattering:

1. Preetham, Shirley, Smits
2. Schafhitzel, Falk, Ertl
3. Bruneton, Neyret
4. Riley, Ebert, Kraus, Tessendorf, Hansen
5. Hoffman, Preetham
6. Nishita, Sirai, Tadamura, Nakamae
7. Nielsen, Christensen

These papers are being referenced in the code in the following way:

```
See [N] page P equation (E)
```

Where the letters are being replaced according to this:

- N: Paper #
- P: Page #
- E: Equation # (if available)

1.15 Changelog

```
VERSION 2.0.9
-------------

- Fixed time not getting incremented properly
- Fixed inaccuracies when progressing time and moon phase with extremely high frame rates
- Fixed inaccuracies when progressing time and moon phase with extremely fast time scales

VERSION 2.0.8
-------------

- Fixed that sun and moon could visibly pop in and out if scaled extremely huge
- Fixed that the date would not get fully incremented for extremely fast time scales
- Fixed that the sun shafts could go through clouds
- Tweaked the TOD_Sky.IsDay and TOD_Sky.IsNight thresholds
- Replaced TOD_Time.UpdateInterval with TOD_Sky.Light.UpdateInterval (now only affects the light source)
- Prepared parts of the codebase for Unity 5 (specifically the new transform behaviour)

VERSION 2.0.7
-------------

- Fixed an issue where the ambient light color would never fully lerp to the night value

VERSION 2.0.6
-------------

- Replaced Day/Night.AmbientIntensity with Day/Night.AmbientColor to offer more customization options
- Added Light.AmbientColoring to adjust ambient light coloring at dusk and dawn
- Added example scripts to enable / disable lights in the scene at day / night / weather
- Added inspector variable to adjust the time update interval in TOD_Time
- Added option to use the real-life moon position rather than the fake "opposite to sun" moon position
- Made all components of TOD_Sky initialize before Start() so that they are accessible from other scripts
- Disabled the automatic light source shadow type adjustment so that the user can manually set it

VERSION 2.0.5
```

```
-------------

- Changed cloud scale parameters from float to 2D vectors to define different scales in x and y direction
- Fixed TOD_Camera always causing the scene to be edited if enabled
- Fixed cloud inconsistencies between linear and gamma color space
- Fixed moon halo disappearing in gamma color space and made the color alpha affect its visibility
- Fixed an issue where the demo mouse look script could overwrite previously imported Standard Assets
- Fixed possible sun and moon gimbal lock that could cause them to spin towards zenith
- Fixed sun shafts being too faint in some setups
- Improved overall lighting calculations
- Improved moon visuals
- Made the sky dome play nice with "depth only" clear flags
- Made the cloud coloring still darken the clouds even for very low values
- Made Components.Animation.CloudUV modulo with the cloud scale to avoid unnecessarily large values
- Added inspector variables to adjust sun shaft base color and sun shaft coloring
- Added the property Cycle.Ticks to get the time information as a long for easy network synchronization
- Added the property Cycle.DateTime to get the time information as a System.DateTime
- Added an inspector variable to set a minimum value for the light source height

VERSION 2.0.4
-------------

- Added a property for the atmosphere renderer component to TOD_Components
- Added properties for all child mesh filter components to TOD_Components
- Changed the quality settings to be adjustable at runtime via public enum inspector variables
- Merged the three prefabs into a single prefab as separate quality prefabs are no longer required
- Fixed the materials always showing up in version control
- Fixed the sky dome always causing the scene to be modified and the editor always asking to save on close
- Fixed the customized sky dome inspector not always looking like the default inspector
- Improved the performance of all cloud shaders by reducing interpolations from frag to vert
- Improved the visuals of all cloud shaders and streamlined their style
- Increased the default cloud texture import resolution to 1024x1024
- Added a white noise texture for future use

VERSION 2.0.3
-------------

- Fixed all issues with DX11 rendering in order to fully support DX11 from this point on

VERSION 2.0.2
-------------

- Fixed an issue where the image effect shaders could overwrite previously imported Standard Assets

VERSION 2.0.1
-------------

- Changed date and time organization to represent the valid Gregorian calendar
- Addressed issues with the Unity sun shaft image effect by providing a modified image effect
- Fixed clouds not correctly handling the planetary atmosphere curvature
- Fixed clouds not offsetting according to the world position of the sky dome
- Fixed cloud glow passing through even the thickest of clouds
- Fixed cloud shadow projection
- Fixed Light.Falloff not affecting the toggle point of the light position between sun and moon
- Automatically disable the corresponding shadows if Day/Night/Clouds.ShadowStrength is set to 0
- Removed Clouds.ShadowProjector toggle as it is no longer required
- Tweaked the old moon halo to not require an additional draw call and added it back in
- Made the sky dome position in world space add an offset to the cloud UV coordinates
- Added Light.Coloring to adjust the light coloring separate from the sky coloring
- Rescaled some parameters for easier use and tweaked their default values

VERSION 2.0.0
-------------

- Moved all documentation to Doxygen
- Renamed the folder "Sky Assets" to "Assets"
- Made the color space be detected automatically by default
- Reworked the sun texture and shader
- Allow light source intensities greater than one
- Reworked the way ambient light is being calculated
- Reworked the way light affects the atmosphere and clouds
- Improved all scattering calculations, especially the integral approximation
- Automatically disable space the game object at night
```

```
- Added a public method to sample the sky dome color in any viewing direction
- Added a fog bias parameter to lerp between zenith and horizon color
- Adjusted the atmosphere alpha calculation
- Added a parameter to easily adjust the scattering color
- Added shader parameters for the moon texture color and contrast
- Adjusted the render queue positions
- Removed the moon halo material as it is no longer required
- Added the physical scattering model to the night sky
- Greatly improved the weather system
- Added fog and contrast parameters to the atmosphere
- Restructured the parameter classes to be more intuitive to use
- Moved all component references into a separate class
- Made the sky presets be applied via editor script rather than separate prefabs
- Improved cloud shading and performance across the board
- Removed the cloud shading parameter
- Added cloud glow from the sun and moon
- Added sky and cloud tone multipliers to sun and moon
- Added viewer height and horizon offset parameters
- Slightly improved overall performance
- Replaced ambient intensity with two parameters for sun and moon
- Replaced the two directional lights with a single one that automatically follows either sun or moon

VERSION 1.7.3
-------------

- Added two parameters "StarTiling" and "StarDensity" to the "Night" section
- Added "Offset -1, -1" to the cloud shadow shaders to avoid Z-fighting on some platforms
- Tweaked the cloud shader for more consistent results in linear and gamma color space
- Tweaked the moon texture to be a lot brighter by default, especially on mobile
- Tweaked the automatically calculated fog color to be similar to the horizon color
- Removed the property "Brightness" from the moon shader as it is no longer needed

VERSION 1.7.2
-------------

- Fixed the ambient light calculation being too dark, even with high ambient light parameter values
- Added the properties "SunZenith" and "MoonZenith" to access sun and moon zenith angles in degrees
- Added a paramter "Halo" to adjust the moon halo intensity and made its color be derived from the light
- Changed several parameters to be clamped between 0 and 1
- Changed the name of the property "OrbitRadius" to "Radius"
- Tweaked the moon phase calculation of both moon mesh and moon halo
- Tweaked several default parameter values of the prefabs

VERSION 1.7.1
-------------
- Changed the default cardinal direction axes of the sky dome (x axis is now west/east, z axis south/north)
- Removed the property "ZenithFactor" as it is no longer being used
- Moved all child object references into a separate toggleable section called "Children"
- Tweaked the default parameters of the prefabs (brightness, haziness, cloud color, moon light intensity)
- Tweaked the calculations of the moon light color, ambient light at night and cloud tone at night
- Tweaked the default sun and moon base color based on good real life approximations
- Tweaked the moon halo
- Renamed the parameter "ShadowAlpha" in "Clouds" to "ShadowStrength"
- Added the parameter "ShadowStrength" for the sun and moon lights

VERSION 1.7.0
-------------

- Fixed an issue where the sun could incorrectly travel around the north,
  even though the location is in the northern hemisphere (Thanks Gregg!)
- Fixed an issue that led to the brightest parts of the sky dome being slightly too dark
- Fixed the automatically calculated fog color not being exactly the same as the horizon
- Added a name prefix to all components to prevent name collisions with other packages
- Added cloud shadows (can be disabled)
- Added UTC time zone support
- Added a parameter to configure the color of the light reflected by the moon
- Added parameters for wind direction in degrees and wind speed in knots
- Added an option to automatically adjust the ambient light color (disabled by default)
- Added a parameter to adjust the sun's light color
- Added a plane with an additive shader at the sun's position to always render a circular sun
- Added dynamic cloud shape adjustments to the "Low" prefab (cloud weather types will now also work)
- Added shading calculations to the "Low" and "Medium" prefabs
- Improved the performance of "Low" prefab by reducing the vertex count
```

```
- Improved the performance of "Low" prefab by removing the moon halo for that prefab by default
- Improved the cloud shading of the "High" prefab
- Improved the visual quality of the weather presets
- Improved the calculation of the sun's position
- Changed the automatic fog color adjustment to be disabled by default
- Changed the moon halo to adjust according to the moon phase
- Changed the name of the parameter from "Color" to "AdditiveColor" for both day and night
- Changed the cloud animation to support network synchronization
- Changed the default tiling of the stars texture to 1 (was 3)
- Changed the moon vertex count in all presets to scale with the device performance
- Removed the parameter "CloudColor" from "NightParameters" as it is now derived from the moon light color

VERSION 1.6.1
-------------
- Fixed an issue related to HDR rendering

VERSION 1.6.0
-------------

- Improved the visuals and functionality of the weather system
  (most METAR codes should now be possible to achieve visually)
- Improved performance of the moon halo shader
- Added official support for HDR rendering
- Replaced the sun mesh with implicit sun scattering in the atmosphere layer
  to reduce dome vertex count, draw calls and pixel overdraw
- Added an additional quality level (now Low/Medium/High instead of Desktop/Mobile)
- Added sky dome presets from various locations around the globe for easier use
- Tweaked the wavelength constants a little to allow for a wider range of sun coloring adjustments

VERSION 1.5.1
-------------

- Fixed an issue causing a missing sun material in the mobile prefab

VERSION 1.5.0
-------------

- Enabled mip mapping of the stars texture by default to avoid flickering
- Added support for using custom skyboxes at night (see readme for details)
- Greatly improved the parametrization of the sun color influence at sunrise and sunset
- Added internal pointers to commonly used components for faster access
- Split the sun and moon parameters into their own property classes
- Adjusted the cloud shading calculation to keep it from darkening some clouds too much
- Adjusted the color wavelengths to produce a more realistic blue color of the sky by default
- Made the moon phase influence the intensity of the sunlight reflected by the moon
- Replaced the lens flares with custom halo shaders that are correctly being occluded by clouds
- Enabled the new halo effects on mobile
- Moved all shaders into a "Time of Day" category
- Added a basic weather manager with three weather types

VERSION 1.4.0
-------------

- Added "Fog { Mode Off }" to the shaders to properly ignore fog
- Added the parameter "Night Cloud Color" to render clouds at night
- Added the parameter "Night Haze Color" to render some haze at night
- Added the parameter "Night Color" to add some color to the night sky
- Renamed the parameter "Haze" to "Haziness"
- Renamed the parameter "Sky Tone" to "Brightness"
- Renamed the properties "Day" and "Night" to "IsDay" and "IsNight"
- Restructured all sky parameters into groups
- Improved the sun lens flare texture
- Improved the stars texture
- Fixed a rendering artifact at the horizon for low haziness values
- Made the scattering calculation in gamma space look identical to linear space

VERSION 1.3.0
-------------

- Greatly improved performance on mobile devices
- Greatly improved sunset and sunrise visual quality
- Added a parameter to control how strongly the sun color affects the sky color
- Added realistic sun and moon lens flare effects
```

```
- Added two additional cloud noise textures
- Improved handling of latitude and longitude
- Made the sky dome render correctly independent of its rotation

VERSION 1.2.0
-------------

- Fixed some bugs regarding linear vs. gamma space rendering
- Fixed some issues with the horizon fadeout
- Adjusted sun and moon size
- Optimized sun and fog color calculation
- Greatly improved visual quality of the cloud system
- Added parameter to control cloud tone, allowing for dark clouds
- Added improved stars texture at night
- Added parameter to control the sun color falloff speed

VERSION 1.1.0
-------------

- First public release on the Asset Store

VERSION 1.0.0
-------------

- First private release for internal use
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Class Documentation

4.1 TOD_Animation Class Reference

Cloud animation class.

Public Attributes

- float WindDegrees = 0.0f

  Wind direction in degrees. = 0 for wind blowing in northern direction. = 90 for wind blowing in eastern direction. = 180 for wind blowing in southern direction. = 270 for wind blowing in western direction.

- float WindSpeed = 3.0f

  Speed of the wind that is acting upon the clouds.

Properties

- Vector4 CloudUV `[get, set]`

  Current cloud UV coordinates. Can be synchronized between multiple game clients to guarantee identical cloud positions.

- Vector4 OffsetUV `[get]`

  Current offset UV coordinates. Is being calculated from the sky dome world position.

4.1.1 Detailed Description

Cloud animation class.

Component of the sky dome parent game object.

The documentation for this class was generated from the following file:

- TOD_Animation.cs

4.2 TOD_AtmosphereParameters Class Reference

Parameters of the atmosphere.

Public Member Functions

- void CheckRange ()

  Assures that all parameters are within a reasonable range.

Public Attributes

- Color ScatteringColor = Color.white

  Artistic value to shift the scattering color of the atmosphere. Can be used to easily simulate alien worlds.
- float RayleighMultiplier = 1.0f

  [0, ∞] Intensity of the atmospheric Rayleigh scattering. Generally speaking this resembles the static scattering.
- float MieMultiplier = 1.0f

  [0, ∞] Intensity of the atmospheric Mie scattering. Generally speaking this resembles the angular scattering.
- float Brightness = 1.0f

  [0, ∞] Brightness of the atmosphere. This is being applied as a simple multiplier to the output color.
- float Contrast = 1.0f

  [0, ∞] Contrast of the atmosphere. This is being applied as a power of the output color.
- float Directionality = 0.5f

  [0, 1] Directionality factor that determines the size and sharpness of the glow around the light source.
- float Haziness = 0.5f

  [0, 1] Intensity of the haziness of the sky at the horizon.
- float Fogginess = 0.0f

  [0, 1] Density of the fog covering the sky. This does not affect the RenderSettings fog that is being applied to other objects in the scene.

4.2.1 Detailed Description

Parameters of the atmosphere.

The documentation for this class was generated from the following file:

- TOD_Parameters.cs

4.3 TOD_Camera Class Reference

Camera class.

Public Attributes

- TOD_Sky sky

  Sky dome reference inspector variable. Has to be manually set to the sky dome instance.
- bool DomePosToCamera = true

  Inspector variable to automatically move the sky dome to the camera position in OnPreCull().
- bool DomeScaleToFarClip = false

  Inspector variable to automatically scale the sky dome to the camera far clip plane in OnPreCull().
- float DomeScaleFactor = 0.95f

  Inspector variable to adjust the sky dome scale factor relative to the camera far clip plane.

4.3.1 Detailed Description

Camera class.

Component of the main camera of the scene to move and scale the sky dome.

The documentation for this class was generated from the following file:

- TOD_Camera.cs

4.4 TOD_CloudParameters Class Reference

Parameters of the cloud layers.

Public Member Functions

- void CheckRange ()

  Assures that all parameters are within a reasonable range.

Public Attributes

- float Density = 3.0f

  [0, ∞] Density multiplier of the cloud layer.
  = 0 no clouds.
  > 0 thicker clouds that are less transparent.
- float Sharpness = 3.0f

  [0, ∞] Sharpness multiplier of the cloud layer.
  = 0 one giant cloud.
  > 0 several smaller clouds.
- float Brightness = 1.0f

  [0, ∞] Brightness multiplier of the cloud layer.
  = 0 black clouds.
  > 0 brighter clouds.
- float ShadowStrength = 0f

  [0, 1] Opacity of the cloud shadows.
- Vector2 Scale1 = new Vector2(3, 3)

  [1, ∞] Scale of the first cloud layer.
- Vector2 Scale2 = new Vector2(7, 7)

  [1, ∞] Scale of the second cloud layer.

4.4.1 Detailed Description

Parameters of the cloud layers.

The documentation for this class was generated from the following file:

- TOD_Parameters.cs

4.5 TOD_Components Class Reference

Component manager class.

Public Member Functions

- void Initialize ()

  Initializes all component references.

Public Attributes

- GameObject Sun = null

  Sun child game object reference.
- GameObject Moon = null

  Moon child game object reference.
- GameObject Atmosphere = null

  Atmosphere child game object reference.
- GameObject Clear = null

  Clear child game object reference.
- GameObject Clouds = null

  Clouds child game object reference.
- GameObject Space = null

  Space child game object reference.
- GameObject Light = null

  Light child game object reference.
- GameObject Projector = null

  Projector child game object reference.
- Transform DomeTransform

  Transform component of the sky dome game object.
- Transform SunTransform

  Transform component of the sun game object.
- Transform MoonTransform

  Transform component of the moon game object.
- Transform CameraTransform

  Transform component of the main camera game object.
- Transform LightTransform

  Transform component of the light source game object.
- Renderer SpaceRenderer

  Renderer component of the space game object.
- Renderer AtmosphereRenderer

  Renderer component of the atmosphere game object.
- Renderer ClearRenderer

  Renderer component of the clear game object.
- Renderer CloudRenderer

  Renderer component of the cloud game object.
- Renderer SunRenderer

  Renderer component of the sun game object.
- Renderer MoonRenderer

  Renderer component of the moon game object.
- MeshFilter SpaceMeshFilter

  MeshFilter component of the space game object.
- MeshFilter AtmosphereMeshFilter

  MeshFilter component of the atmosphere game object.
- MeshFilter ClearMeshFilter

  MeshFilter component of the clear game object.

- MeshFilter CloudMeshFilter

  MeshFilter component of the cloud game object.
- MeshFilter SunMeshFilter

  MeshFilter component of the sun game object.
- MeshFilter MoonMeshFilter

  MeshFilter component of the moon game object.
- Material SpaceShader

  Main material of the space game object.
- Material AtmosphereShader

  Main material of the atmosphere game object.
- Material ClearShader

  Main material of the clear game object.
- Material CloudShader

  Main material of the cloud game object.
- Material SunShader

  Main material of the sun game object.
- Material MoonShader

  Main material of the moon game object.
- Material ShadowShader

  Main material of the projector game object.
- Light LightSource

  Light component of the light source game object.
- Projector ShadowProjector

  Projector component of the shadow projector game object.
- TOD_Sky Sky

  Sky component of the sky dome game object.
- TOD_Animation Animation

  Animation component of the sky dome game object.
- TOD_Time Time

  Time component of the sky dome game object.
- TOD_Weather Weather

  Weather component of the sky dome game object.
- TOD_Resources Resources

  Resource container component of the sky dome game object.
- TOD_SunShafts SunShafts

  Sun shaft component of the camera game object if available.

4.5.1 Detailed Description

Component manager class.

Component of the main camera of the scene.

The documentation for this class was generated from the following file:

- TOD_Components.cs

4.6 TOD_CycleParameters Class Reference

Parameters of the day and night cycle.

Public Types

- enum MoonPositionType { **OppositeToSun**, **Realistic** }

  Available methods to calculate the moon position.

Public Member Functions

- void CheckRange ()

  Assures that all parameters are within a reasonable range.

Public Attributes

- float Hour = 12

  [0, 24] Time of the day in hours.
  = 0 at the start of the day.
  = 12 at noon.
  = 24 at the end of the day.
- int Day = 1

  [1, 28-31] Current day of the month.
- int Month = 3

  [1, 12] Current month of the year.
- int Year = 2000

  [1, 9999] Current year.
- float MoonPhase = 0.0f

  [-1, 1] Phase of the moon.
  = 0 full moon.
  ± 1 no moon.
- MoonPositionType MoonPosition = MoonPositionType.OppositeToSun

  Type of the moon position calculation.
- float Latitude = 0f

  [-90, 90] Latitude of your position in degrees.
  = -90 at the south pole.
  = 0 at the equator.
  = 90 at the north pole.
- float Longitude = 0f

  [-180, 180] Longitude of your position in degrees.
  = -180 at 180 degrees in the west of Greenwich, England.
  = 0 at Greenwich, England.
  = 180 at 180 degrees in the east of Greenwich, England.
- float UTC = 0f

  UTC/GMT time zone of the current location.
  = 0 for Greenwich, England.

Properties

- System.DateTime DateTime `[get, set]`

  All time information as a System.DateTime instance.
- long Ticks `[get, set]`

  All time information as a single long. Value corresponds to the System.DateTime.Ticks property.

4.6.1 Detailed Description

Parameters of the day and night cycle.

The documentation for this class was generated from the following file:

- TOD_Parameters.cs

4.7 TOD_DayParameters Class Reference

Parameters that are unique to the day.

Public Member Functions

- void CheckRange ()

  Assures that all parameters are within a reasonable range.

Public Attributes

- Color AdditiveColor = Color.black

  Artistic value for an additive color at day.
- Color AmbientColor = new Color32(255, 243, 234, 200)

  Color of the ambient light at day. TOD_WorldParameters.SetAmbientLight has to be set for this to have any effect.
- Color SunLightColor = new Color32(255, 243, 234, 255)

  Color of the light emitted by the sun.
- Color SunMeshColor = new Color32(255, 233, 180, 255)

  Color of the sun material.
- Color SunShaftColor = new Color32(255, 243, 234, 255)

  Color of the sun shafts cast by the sun.
- float SunMeshSize = 1.0f

  [0, ∞] Size of the sun mesh in degrees.
- float SunLightIntensity = 0.75f

  [0, ∞] Intensity of the sun light source.
- float ShadowStrength = 1.0f

  [0, 1] Opacity of the object shadows dropped by the sun light source.
- float SkyMultiplier = 1.0f

  [0, 1] Sky opacity multiplier at day.
- float CloudMultiplier = 1.0f

  [0, 1] Cloud tone multiplier at day.

4.7.1 Detailed Description

Parameters that are unique to the day.

The documentation for this class was generated from the following file:

- TOD_Parameters.cs

4.8 TOD_LightParameters Class Reference

Parameters of the light source.

Public Member Functions

- void CheckRange ()

  Assures that all parameters are within a reasonable range.

Public Attributes

- float UpdateInterval = 0

  Light source position update interval in seconds. Zero means every frame.
- float MinimumHeight = 0.0f

  [0, 1] Controls how low the light source is allowed to go.
  = -1 light source can go as low as it wants.
  = 0 light source will never go below the horizon.
  = +1 light source will never leave zenith.
- float Falloff = 0.7f

  [0, 1] Controls how fast the sun color falls off. This is especially visible during sunset and sunrise.
- float Coloring = 0.7f

  [0, 1] Controls how strongly the light color is being affected by sunset and sunrise.
- float SkyColoring = 0.5f

  [0, 1] Controls how strongly the sun color affects the atmosphere color. This is especially visible during sunset and sunrise.
- float CloudColoring = 0.9f

  [0, 1] Controls how strongly the sun color affects the cloud color. This is especially visible during sunset and sunrise.
- float ShaftColoring = 0.9f

  [0, 1] Controls how strongly the sun shaft color is being affected by sunset and sunrise.
- float AmbientColoring = 0.5f

  [0, 1] Controls how strongly the ambient color is being affected by sunset and sunrise.

4.8.1 Detailed Description

Parameters of the light source.

The documentation for this class was generated from the following file:

- TOD_Parameters.cs

4.9 TOD_NightParameters Class Reference

Parameters that are unique to the night.

Public Member Functions

- void CheckRange ()

  Assures that all parameters are within a reasonable range.

Public Attributes

- Color AdditiveColor = Color.black

  Artistic value for an additive color at night.
- Color AmbientColor = new Color32(181, 204, 255, 50)

  Color of the ambient light at night. TOD_WorldParameters.SetAmbientLight has to be set for this to have any effect.

- Color MoonLightColor = new Color32(181, 204, 255, 255)

  Color of the light emitted by the moon.
- Color MoonMeshColor = new Color32(255, 233, 200, 255)

  Color of the moon material.
- Color MoonHaloColor = new Color32(81, 104, 155, 255)

  Color of the moon halo.
- float MoonMeshSize = 1.0f

  [0, ∞] Size of the moon mesh in degrees.
- float MoonLightIntensity = 0.1f

  [0, ∞] Intensity of the moon light source.
- float ShadowStrength = 1.0f

  [0, 1] Opacity of the object shadows dropped by the moon light source.
- float SkyMultiplier = 0.1f

  [0, 1] Sky opacity multiplier at night.
- float CloudMultiplier = 0.2f

  [0, 1] Cloud tone multiplier at night.

4.9.1 Detailed Description

Parameters that are unique to the night.

The documentation for this class was generated from the following file:

- TOD_Parameters.cs

4.10 TOD_PostEffectsBase Class Reference

Post effects base class.

4.10.1 Detailed Description

Post effects base class.

Based on the post effects base from the default Unity image effects.

The documentation for this class was generated from the following file:

- TOD_PostEffectsBase.cs

4.11 TOD_Resources Class Reference

Material and mesh wrapper class.

Public Attributes

- Mesh **Quad**
- Mesh **SphereHigh**
- Mesh **SphereMedium**
- Mesh **SphereLow**
- Mesh **IcosphereHigh**
- Mesh **IcosphereMedium**

- Mesh **IcosphereLow**
- Mesh **HalfIcosphereHigh**
- Mesh **HalfIcosphereMedium**
- Mesh **HalfIcosphereLow**
- Material **CloudMaterialBumped**
- Material **CloudMaterialDensity**
- Material **CloudMaterialFastest**
- Material **ShadowMaterialBumped**
- Material **ShadowMaterialDensity**
- Material **ShadowMaterialFastest**
- Material **SpaceMaterial**
- Material **AtmosphereMaterial**
- Material **SunMaterial**
- Material **MoonMaterial**
- Material **ClearMaterial**

4.11.1 Detailed Description

Material and mesh wrapper class.

Component of the sky dome parent game object.

The documentation for this class was generated from the following file:

- TOD_Resources.cs

4.12 TOD_Sky Class Reference

Main sky dome management class.

Public Types

- enum ColorSpaceDetection { **Auto**, **Linear**, **Gamma** }

  Available methods to detect the Unity color space.
- enum CloudQualityType { **Fastest**, **Density**, **Bumped** }

  Available methods to render the clouds.
- enum MeshQualityType { **Low**, **Medium**, **High** }

  Available vertex count levels for the meshes.

Public Member Functions

- Vector3 OrbitalToUnity (float radius, float theta, float phi)

  Convert spherical coordinates to cartesian coordinates.
- Vector3 OrbitalToLocal (float theta, float phi)

  Convert spherical coordinates to cartesian coordinates.
- Color SampleAtmosphere (Vector3 direction, bool clampAlpha=true)

  Sample atmosphere colors from the sky dome.

Public Attributes

- ColorSpaceDetection UnityColorSpace = ColorSpaceDetection.Auto

  Inspector variable to adjust the color space. Should stay at ColorSpaceDetection.Auto in most cases.
- CloudQualityType CloudQuality = CloudQualityType.Bumped

  Inspector variable to adjust the cloud quality.
- MeshQualityType MeshQuality = MeshQualityType.High

  Inspector variable to adjust the mesh quality.
- TOD_CycleParameters Cycle

  Inspector variable containing parameters of the day and night cycle.
- TOD_AtmosphereParameters Atmosphere

  Inspector variable containing parameters of the atmosphere.
- TOD_DayParameters Day

  Inspector variable containing parameters of the day.
- TOD_NightParameters Night

  Inspector variable containing parameters of the night.
- TOD_LightParameters Light

  Inspector variable containing parameters of the light source.
- TOD_StarParameters Stars

  Inspector variable containing parameters of the stars.
- TOD_CloudParameters Clouds

  Inspector variable containing parameters of the cloud layers.
- TOD_WorldParameters World

  Inspector variable containing parameters of the world.

Properties

- TOD_Components Components `[get, set]`

  Containins references to all components.
- bool IsDay `[get]`

  Boolean to check if it is day.
- bool IsNight `[get]`

  Boolean to check if it is night.
- float Radius `[get]`

  Radius of the sky dome.
- float Gamma `[get]`

  Gamma value that is being used in the shaders.
- float OneOverGamma `[get]`

  Inverse of the gamma value (1 / Gamma) that is being used in the shaders.
- float LerpValue `[get, set]`

  Falls off the darker the sunlight gets. Can for example be used to lerp between day and night values in shaders.
  = +1 at day
  = 0 at night.
- float SunZenith `[get, set]`

  Sun zenith angle in degrees.
  = 0 if the sun is exactly at zenith.
  = 180 if the sun is exactly below the ground.
- float MoonZenith `[get, set]`

  Moon zenith angle in degrees.
  = 0 if the moon is exactly at zenith.
  = 180 if the moon is exactly below the ground.
- float LightZenith `[get]`

Currently active light source zenith angle in degrees.
= 0 if the currently active light source (sun or moon) is exactly at zenith.
= 90 if the currently active light source (sun or moon) is exactly at the horizon.

- float LightIntensity `[get]`

  Current light intensity. Returns the intensity of TOD_Sky.LightSource.
- Vector3 MoonDirection `[get]`

  Moon direction vector in world space. Returns the forward vector of TOD_Sky.MoonTransform.
- Vector3 SunDirection `[get]`

  Sun direction vector in world space. Returns the forward vector of TOD_Sky.SunTransform.
- Vector3 LightDirection `[get]`

  Current directional light vector in world space. Lerps between TOD_Sky.SunDirection and TOD_Sky.MoonDirection at dusk and dawn.
- Color LightColor `[get]`

  Current light color.
- Color SunShaftColor `[get, set]`

  Current sun shaft color.
- Color SunColor `[get, set]`

  Current sun color.
- Color MoonColor `[get, set]`

  Current moon color.
- Color MoonHaloColor `[get, set]`

  Current moon halo color.
- Color CloudColor `[get, set]`

  Current cloud color.
- Color AdditiveColor `[get, set]`

  Current additive color.
- Color AmbientColor `[get, set]`

  Current ambient color.
- Color FogColor `[get]`

  The fog color sampled from the physical model. Depends on camera view direction. This property is O(1) if TOD_WorldParameters.SetFogColor is enabled as its value will be calculated anyhow.

4.12.1 Detailed Description

Main sky dome management class.

Component of the sky dome parent game object.

4.12.2 Member Function Documentation

4.12.2.1 Vector3 TOD_Sky.OrbitalToLocal (float *theta*, float *phi*) `[inline]`

Convert spherical coordinates to cartesian coordinates.

Parameters

theta	Spherical coordinates theta.
phi	Spherical coordinates phi.

Returns

Unity position in local space.

4.12.2.2 Vector3 TOD_Sky.OrbitalToUnity (float *radius,* float *theta,* float *phi*) [inline]

Convert spherical coordinates to cartesian coordinates.

Parameters

radius	Spherical coordinates radius.
theta	Spherical coordinates theta.
phi	Spherical coordinates phi.

Returns

Unity position in world space.

4.12.2.3 Color TOD_Sky.SampleAtmosphere (Vector3 *direction*, bool *clampAlpha* = `true`) [inline]

Sample atmosphere colors from the sky dome.

Parameters

direction	View direction in world space.

Returns

Color of the atmosphere in the specified direction.

The documentation for this class was generated from the following files:

- TOD_Sky.cs
- TOD_Sky_Variables.cs
- TOD_Sky_Quality.cs
- TOD_Sky_Unity.cs

4.13 TOD_StarParameters Class Reference

Parameters of the stars.

Public Member Functions

- void CheckRange ()

  Assures that all parameters are within a reasonable range.

Public Attributes

- float Tiling = 2.0f

  [0, ∞] Texture tiling of the stars texture. Determines how often the texture is tiled accross the sky and therefore the size of the stars.
- float Density = 0.5f

  [0, 1] Amount of stars that are visible.

4.13.1 Detailed Description

Parameters of the stars.

The documentation for this class was generated from the following file:

- TOD_Parameters.cs

4.14 TOD_SunShafts Class Reference

Sun shaft class.

Public Types

- enum SunShaftsResolution { **Low**, **Normal**, **High** }

  Available resolutions for the sun shafts. High is full, Normal is half and Low is quarter the screen resolution.
- enum SunShaftsBlendMode { **Screen**, **Add** }

  Available methods to blend the sun shafts with the image.

Public Attributes

- TOD_Sky sky = null

  Sky dome reference inspector variable. Has to be manually set to the sky dome instance.
- SunShaftsResolution Resolution = SunShaftsResolution.Normal

  Inspector variable to define the sun shaft rendering resolution.
- SunShaftsBlendMode BlendMode = SunShaftsBlendMode.Screen

  Inspector variable to define the sun shaft rendering blend mode.
- int RadialBlurIterations = 2

  Inspector variable to define the number of blur iterations to be performaed.
- float SunShaftBlurRadius = 2

  Inspector variable to define the radius to blur filter applied to the sun shafts.
- float SunShaftIntensity = 1

  Inspector variable to define the intensity of the sun shafts.
- float MaxRadius = 0.5f

  Inspector variable to define the maximum radius of the sun shafts.
- bool UseDepthTexture = true

  Inspector variable to define whether or not to use the depth buffer. If enabled, requires the target platform to allow the camera to create a depth texture. Unity always creates this depth texture if deferred lighting is enabled. Otherwise this script will enable it for the camera it is attached to. If disabled, requires all shaders writing to the depth buffer to also write to the frame buffer alpha channel. Only the frame buffer alpha channel will then be used to check for shaft blockers in the image effect. This is being done correctly in the built-in Unity opaque shaders as they always write 1 to the alpha channel. However, this is not being done in most of the built-in Unity cutout transparent and terrain shaders.
- Shader SunShaftsShader = null

  Inspector variable pointing to the sun shaft rendering shader.
- Shader ScreenClearShader = null

  Inspector variable pointing to the clear rendering shader.

4.14.1 Detailed Description

Sun shaft class.

Component of the main camera of the scene to render sun shafts. Based on the sun shafts from the default Unity image effects.

The documentation for this class was generated from the following file:

- TOD_SunShafts.cs

4.15 TOD_Time Class Reference

Time iteration class.

Public Attributes

- float DayLengthInMinutes = 30

  Day length inspector variable. Length of one day in minutes.
- bool ProgressTime = true

  Time progression inspector variable. Automatically updates Cycle.Hour if enabled.
- bool ProgressDate = true

  Date progression inspector variable. Automatically updates Cycle.Day if enabled.
- bool ProgressMoonPhase = true

  Moon phase progression inspector variable. Automatically updates Moon.Phase if enabled.

4.15.1 Detailed Description

Time iteration class.

Component of the sky dome parent game object.

The documentation for this class was generated from the following file:

- TOD_Time.cs

4.16 TOD_Weather Class Reference

Weather management class.

Public Types

- enum CloudType {
  Custom, **None**, **Few**, **Scattered**,
  Broken, **Overcast** }

  Available cloud coverage types.
- enum WeatherType {
  Custom, **Clear**, **Storm**, **Dust**,
  Fog }

  Available weather types.

Public Attributes

- float FadeTime = 10f

  Fade time inspector variable. Time to fade from one weather type to the other.
- CloudType Clouds = CloudType.Custom

  Currently selected CloudType.
- WeatherType Weather = WeatherType.Custom

  Currently selected WeatherType.

4.16.1 Detailed Description

Weather management class.

Component of the sky dome parent game object.

The documentation for this class was generated from the following file:

- TOD_Weather.cs

4.17 TOD_WorldParameters Class Reference

Parameters affecting other objects of the world.

Public Member Functions

- void CheckRange ()

  Assures that all parameters are within a reasonable range.

Public Attributes

- bool SetAmbientLight = false

  Automatically adjust the ambient light color of your render settings.
- bool SetFogColor = false

  Automatically adjust the fog color of your render settingstings.
- float FogColorBias = 0.0f

  [0, 1] Fog color sampling height. TOD_WorldParameters.SetFogColor has to be set for this to have any effect.
  = 0 fog is atmosphere color at horizon.
  = 1 fog is atmosphere color at zenith.
- float ViewerHeight = 0.0f

  [0, 1] Relative viewer height in the atmosphere.
  = 0 on the ground.
  = 1 at the border of the atmosphere.
- float HorizonOffset = 0.0f

  [0, 1] Relative horizon offset.
  = 0 horizon exactly in the middle of the sky dome sphere.
  = 1 horizon exactly at the bottom of the sky dome sphere.

4.17.1 Detailed Description

Parameters affecting other objects of the world.

The documentation for this class was generated from the following file:

- TOD_Parameters.cs

Index