Sky Rendering

Description

Sky rendering consists out of 2 stages:

1) Sun & Atmosphere rendering
2) Clouds rendering (optional)

Sun & Atmosphere rendering

Sun & Atmosphere rendering can be done in 2 ways:

1) Hoffman-Preetham scattering algorithm
2) Static texture on a customizable sky mesh.

Hoffman-Preetham scattering algorithm

Utilizes scattering methods developed by Hoffman and Preetham:
http://developer.amd.com/media/gpu_assets/ATI-LightScattering.pdf

Static texture

Static 2D texture is mapped on sky dome. User can either pick predefined sky dome mesh, or specify custom one.

Clouds Rendering

Animated clouds are rendered into a texture, with lighting effects applied. Afterwards, these texture is overlayed over background sky dome as a plane. The class that handles the process is called *CloudPlane*.

Associated code

`r3dAtmosphere`	Class that describes sky, fogging and some lighting level features
SkyHoffman_ps.hls	Pixel shader for Hoffman-Preetham based sky renderer
SkyHoffman_vs.hls	Vertex shader for Hoffman-Preetham based sky renderer
SkyHoffmanLQ_ps.hls	Pixel shader for Hoffman-Preetham based sky renderer with reduced instruction count for low end GPUs
SkyStatic_vs.hls	Vertex shader for texture based sky rendering
SkyStatic_ps.hls	Pixel shader for texture based sky rendering
`CloudPlane`	Class that describes a horizontal plane with dynamic clouds rendered in it

Associated source files

r3dAtmosphere.h	`r3dAtmosphere` class header
r3dAtmosphere.cpp	`r3dAtmosphere` class implementation
CloudPlane\Cloud.h	Header and source files of CloudPlane and related classes

CloudPlane\Cloud.cpp	
CloudPlane\CloudGrid.h	
CloudPlane\CloudGrid.cpp	
CloudPlane\CloudPlane.h	
CloudPlane\CloudPlane.cpp	Header and source files of CloudPlane and related classes
CloudPlane\SceneParameter.h	
CloudPlane\SceneParameter.cpp	
CloudPlane\Shader.h	
CloudPlane\Shader.cpp	
CloudPlane\Shaders.h	
CloudPlane\Shaders.cpp	
SkyHoffman_ps.hls	Hoffman sky rendering shaders
SkyHoffman_vs.hls	
SkyHoffmanLQ_ps.hls	
SkyStatic_ps.hls	Static sky rendering shaders
SkyStatic_vs.hls	

class r3dAtmosphere

Description

Manages level sky parameters, their saving and loading. This set of parameters is individual for each level.

This class has the following important fields:

Name	Type	Field
StaticSkyTex	*r3dTexture**	Texture for static (textured) sky rendering
StaticSkyMesh	*r3dMesh**	Mesh for textured sky rendering
StaticSkyTexName	*r3dString*	File name of the static sky texture
StaticSkyMeshName	*r3dString*	File name of the static sky mesh
ParticleShadingCoef	*float*	Shadowing coefficient for particle systems
StaticTexGenScaleX	*float*	In case texture coordinate generation is used for the static sky(*bStaticSkyPlanarMapping != 0)*, this is the texture U scale
StaticTexGenScaleY	*float*	In case texture coordinate generation is used for the static sky (*bStaticSkyPlanarMapping != 0)*, this is the texture V scale
StaticTexGetOffsetX	*float*	In case texture coordinate generation is used for the static sky (*bStaticSkyPlanarMapping != 0)*, this is the texture U offset
StaticTexGetOffsetY	*float*	In case texture coordinate generation is used for the static sky (*bStaticSkyPlanarMapping != 0)*, this is the texture V offset
bStaticSkyEnable	*int*	Controls, whether static (textured) sky is enabled, thus disabling Hoffman calculated sky.
bCustomStaticMeshEn	*int*	Controls, whether static sky uses custom user specified

able		mesh
bStaticSkyPlanarMapping	*int*	Controls if planar texture coordinate generation is to be used for static sky texture coordinates
SunLightOn	*int*	Controls if sun light is enabled on this level. For night levels, the sun light is best disabled, as it greatly increases performance – no sun shadows have to be rendered.
SunColor	*r3dTimeGradient2*	Controls sun color during different times of day. Used for Hoffman scattering shader.
SkyColor	*r3dTimeGradient2*	Controls sky color during different times of day. Used for Hoffman scattering shader.
SunIntensity	*float*	Controls overall intensity of the sun. Used for Hoffman scattering shader.
LambdaCol	*r3dTimeGradient2*	Controls *LambdaCol* hoffman scattering shader param
BacklightColor	*r3dTimeGradient2*	Controls Backlight color for different times of day. Used in deferred sun lighting shader.
BacklightIntensity	*float*	Intensity of the back light
bVolumeFog	*int*	Controls if the volume fog is to be used with this level
Fog_MaxHeight	*float*	Controls max height of the fog.
Fog_Color	*r3dTimeGradient2*	Controls fog color.
Fog_Range	*r3dTimeGradient2*	Controls fog distance.
Fog_Distance	*r3dTimeGradient2*	Controls fog distance.
Fog_Density	*r3dTimeGradient2*	Controls fog density.
Fog_Height	*r3dTimeGradient2*	Controls fog height.
Aerial_Density	*r3dTimeGradient2*	Controls aerial effect density of the fog effect
Aerial_Distance	*r3dTimeGradient2*	Controls aerial effect distance of the fog effect
SkyFog_Start	*float*	Fog start for Hoffman scattering shader
SkyFog_End	*float*	Fog end for Hoffman scattering shader
HGg	*r3dTimeGradient2*	HGg parameter for Hoffman scattering shader
InscatteringMultiplier	*r3dTimeGradient2*	*InscatteringMultiplier* parameter for Hoffman scattering shader
BetaRayMultiplier	*r3dTimeGradient2*	*BetaRayMultiplier* parameter for Hoffman scattering shader
BetaMieMultiplier	*r3dTimeGradient2*	*BetaMieMultiplier* parameter for Hoffman scattering shader
SunIntensityCoef	*r3dTimeGradient2*	*SunIntensityCoef* parameter for Hoffman scattering shader
Turbitity	*r3dTimeGradient2*	*Turbitity* parameter for Hoffman scattering shader
SunElevationAngle	*float*	Sun elevation angle for controlling the sun's "latitude"
SSSDiffuse	*float*	Globally applied subsurface scattering diffuse intensity
SSSSpecular	*float*	Globally applied subsurface scattering specular intensity
SSSSpecPower	*float*	Globally applied subsurface scattering specular power
RainParticleSystem	*GameObject**	Particle system for that level, which moves with the camera
RainParticleSystemName	*char[64]*	File name of the particle system, which moves with the camera

Important methods

```
int LoadFromXML( pugi::xml_node root )
```

Summary:

Loads sky settings from xml node *root*

Parameters:

root – xml node to load settings from

Return value:

Non-zero value if successful, zero value otherwise.

```
int SaveToXML( pugi::xml_node root )
```

Summary:

Saves sky settings to xml node *root*.

Parameters:

root – xml node to save settings to.

Return value:

Non-zero value if successful, zero value otherwise.

```
void ReloadTextures()
```

Summary:

Reloads sky related textures in order to conform with new texture quality settings.

```
void Update()
```

Summary:

Update *r3dAtmosphere* state. Must be called once per frame.

```
void SetRainParticle( const char* Name )
```

Summary:

Binds rain particle to *r3dAtmosphere*. The position of the rain particle is automaticly set to camera position.

Parameters:

Name - file name of the particle to bind.

---

void ClearRainParticle()

Summary:

Removes the rain particle.

---

void EnableStaticSky()

Summary:

Enables rendering using static sky method.

---

void DisableStaticSky()

Summary:

Disables rendering using static sky method. Rendering using Hoffman scattering method is used afterwards.

---

void SetStaticSkyTexture(const r3dString& textureName)

Summary:

Sets static sky texture for static sky rendering method.

Parameters:

textureName - file name of the texture to use for static sky

---

void SetStaticSkyMesh(const r3dString& meshName)

Summary:

Set custom mesh for sky dome to use with static sky rendering method.

Parameters:

meshName - mesh file name to use for static sky rendering method.