Hierarchical Particlesystems Demo Reference Manual

Generated by Doxygen 1.4.6-NO

Thu Apr 27 23:31:27 2006

Chapter 1

Hierarchical Particlesystems Demo Class Documentation

1.1 AdvancedParticleSystem Class Reference

Implements a hierarchical particle system.

Public Member Functions

- void **Display** (Camera ∗cam, Vector LightPos)
  renders the particle system from the given camera using the given light position
- void **Refresh** (Camera ∗cam, Vector LightPos, unsigned int Dt, unsigned int TimefromSecond)
  refreshes the particle system
- void **RefreshDepths** (Camera ∗cam)
  generates front and back depth textures od the smaller particle system
- void **Initialize** ()
  initialize textures and gpu programs
- void **InitSystems** ()
  initialize the two particle systems
- void **RenderObjectDepths** (Camera ∗cam)
  generates scene depth texture
- void **RefreshIllumTextures** (Vector LightPos)
  refreshes the illumination texture

Public Attributes

- float **albedo**

albedo of the particles

- float **symmetry**
  scattering symmetry of the scattering (used in phase function calculation)
- float **transparency**
  transparency of the particles
- ParticleSystem **m_ParticleSystem**
  particle system made ot of a smaller system
- ParticleSystem **m_LittleParticleSystem**
  smaller particle system

Private Attributes

- Impostor **m_DisplayImpostor**
  Impostor to display.
- Impostor **m_IllumImpostor**
  Impostor for illumination calculation.
- RenderTexture **m_IllumTexture**
  Illumination texture.
- RenderTexture **m_FrontDepthTexture**
  Stores nearest and furthest distances of the particles from the camera and also the opacity.
- RenderTexture **m_ObjectsTexture**
  Stores nearest distances of the scene objects from the camera.
- RenderTexture **m_ScatteredIllumTexture**
  A motion blurred version of the illumination texture to avoid abrupt changes.

1.1.1 Detailed Description

Implements a hierarchical particle system.

A hierarchical particle system is a particle system made out of a smaller particle systems. First an image of the smaller particle system should be rendered, than this image can be multiplied and be used as sprite images to form the bigger system.

To avoid incorrect depth culling caused by the simplifications made by considering a group of particles as a single quad we have to take into acount some additional depth information too. We also render the the closest and the furthest depth values of the smaller system in the view camera's space. During display this depth information can be used to estimate the length of the light ray segment travelled in the medium and alter the opacity of the particles respectively.

This particle system implementation also deals with light scattering inside the medium. It builds a layered light absorption texture which stores the amount of absorped light in different depths from lightsource. With

the use of the illumination texture we can approximate self shadowing of the medium. This implementation uses four layers and stores it in the four channels of a texture. This way no 3D textures are needed, and the light absorption (illumination) texture can be generated in a single render pass.

During rendering a mie scattering model is used to compute the amount of scattered light. To speed up calculations phase function values are read back from a 2D look-up texture (phase texture).

1.1.2 Member Function Documentation

1.1.2.1 void AdvancedParticleSystem::RefreshIllumTextures (Vector *LightPos*)

refreshes the illumination texture

The illumination texture is a four layered texture, each layer stored in the separate color channels.