TressFX11

Overview

This sample implements AMD's TressFX hair rendering and simulation technology. The TressFX technology uses the GPU to simulate and render hair that looks and acts like real hair, better than anything previously seen in games.

AMD TressFX makes use of the processing power of high performance GPUs to do realistic rendering and utilizes DirectCompute to physically simulate each individual strand of hair. Rendering is based on an I3D 2012 Paper from AMD "A Framework for Rendering ComplexScattering Effects on Hair", Yang, et.al., with some modifications and optimizations in order to achieve a level of performance that enables TressFX to be used in games. The physics simulation is described in another paper by AMD which was presented at VRIPHYS 2012: "Real-time Hair Simulation with Efficient Hair Style Preservation", Han, et al.

Rendering

The implementation of the rendering part of TressFX is based on three main features required to render good looking hair:

- Antialiasing
- Self-Shadowing
- Transparency

When used with a realistic shading model these features provide realism needed for natural looking hair. The sample uses the well-known Kajiya hair shading model. This uses anisotropic lighting to get the correct kind of highlights that hair has. The hair is rendered as several thousand individual strands stored as thin chains of polygons.

Since the width of a hair is considerably smaller than one pixel Antialiasing is required to achieve good looking results. In TressFX this is done by computing the percentage of pixel coverage for each pixel that is partially covered by a hair.

The second main feature, Self-Shadowing, is necessary for giving the hair a realistic looking texture. Without it, the hair tends to look artificial and more like plastic on a puppet than as opposed to real hair. Shadowing is done using a simplified deep shadow map. Typically a deep shadow map is several layers of depth values. To improve performance and memory usage, the implementation interpolates over a range of depth values to avoid having to keep a list of depth values for each pixel.

Transparency provides a softer look for the hair, similar to real hair. If transparency was not used the strands of hair would look too coarse, especially at the edges. In addition to that real hair is actually translucent, so rendering with transparency is consistent with simulating the lighting properties of real hair. Unfortunately transparent hair is difficult to render because there are thousands of hair strands that need to be sorted. To help with this, TressFX technology uses order independent transparency (OIT). The K-Buffer required for OIT is implemented as a per-pixel linked list (PPLL), so the front-most k hair pixels can be rendered in the correct order. The PPLL is generated by writing into a DX11UAV from the hair pixel shader. Once the PPLL is filled the hair is rendered to the back buffer by applying a full screen quad.

Simulation

Physically accurate simulation of the hair is completely done on the GPU using DirectCompute. The hair responds to movement, wind, and gravity using a Verlet integration method. To maintain shape and natural looking behavior, the simulation uses both local and global shape constraints. Local shape constraints keep individual strands of hair consistent under bending and twisting forces for various types of hair such as straight or curly. Global shape constraints keep the hair in the same hairstyle it was designed for, even after the hair has been temporarily disturbed. Additionally the simulation has length constraints to keep the hair from stretching under forces. A capsule method is used for collision between the hair and the head.

The TressFX hair simulation has many parameters that allow the programmer and artist to modify the look and behavior of the hair. For example the stiffness of the hair can be modified on the fly to make it look wet. The hair can also be made with varying levels of stiffness and damping, also adjustable during execution.

One known problem with the current simulation approach is that if the local/global shape constraints are modeled to look good with gravitation in mind this can result in strange behavior if the gravitation direction is reversed (for example if a character using TressFX is hanging upside down). Should this become necessary it is easiest to create a special hair mesh with simulation parameters authored to look good while upside down.

Authoring

TressFX hair used in the sample and in games has been authored with off the shelf content creation tools. For example, the “Shave and a Haircut” Maya plugin is good for designing hair. The hair can be exported by writing a simple exporter to .tfx format which is used by the TressFX technology. Tfx files are a text files which start with the number of strands and sorting information:

```
numStrands 1595
is sorted 1
```

This is followed by a line which gives the strand number and the number of vertices along with texture coordinates (which can be zero)

```
strand 0 numVerts 11 texcoord 0.000000 000000
```

Immediately following this is x,y,z vertex positions of the strands:

```
-4.80447 103.333 25.4651
-8.97057 105.761 27.8268
-14.103 106.443 29.189
-18.9103 104.199 28.8422
-22.2072 100.107 28.0818
-23.2806 94.8577 27.7387
-23.6553 89.5007 27.697
-24.0455 84.1447 27.6378
-24.3112 78.8325 26.9485
-24.4854 73.8321 25.0157
-24.8791 69.4098 21.9887
```

Since the number of vertices in the strand is 11, the number of coordinates is 11. The specification of strand followed by vertex list continues until all of the strands of hair are specified.

Sample Controls

- Left Mouse Button: rotate the camera around the model.
- Middle Mouse Button: move the model
- Ctrl+ Middle Mouse Button: rotate the model

Render menu options

- Density
  Controls the density of the hair by reducing the number of hairs to be rendered.
- Hair thickness
  Controls thickness of each hair strand
- Strand copies
  Increases the number of hairs to be rendered by duplicating the mesh
- Alpha
  Controls the opaqueness of the hair
- Thin Tip
  If enabled the hairs get thinner towards the tip
- Self Shadowing
  Enables/Disables the shadow rendering
- Shadow Map Alpha
  Controls the intensity of the hair self-shadowing
- Use Coverage
  Enables/Disables computation of per-pixel hair coverage factors
- Use Alternative Coverage
  Enables an alternative coverage computation which is a bit faster
- Change Hair Color
  Opens a Dialog that allows changing the hair color

- Magnify: RMouse
  Toggles if pressing RMouse opens an on screen magnifying region
  - Magnify Region
    Size of the screen region to be magnified in pixels
  - Magnify Scale
    Scaling factor for magnifying region
- Simulation Menu
  Switches from “Render menu” to “Simulation menu”

Simulation menu options

- Simulate
  Enables/Disables hair simulation
- Collision with head
  Enables/Disables collision in the physical simulation
- Hair segment selection control
  Selects which part of the hair is controlled by the 4 sliders below:
  - Damping
    Modifies the damping factor of the hair movement
  - Local stiffness
    Modifies the intensity of the local shape constraints
  - Global shape stiffness
    Modifies the intensity of the global shape constraints
  - Global shape stiffness range
    Modifies how much of the hair is affected by global shape constraints
- Length constraint iterations
  Number of iterations to be computed for length constraints
- Local shape constraint iterations
  Number of iterations to be computed for length constraints
- Wind Magnitude
  Sets the wind magnitude and direction
- Render Menu
  Switches from “Simulation menu” to “Render menu”

AMD Smarter Choice

Advanced Micro Devices
One AMD Place
P.O. Box 3453
Sunnyvale, CA 94088-3453

www.amd.com
http://ati.amd.com/developer