The TressFX Viewer and Authoring

The viewer shows models with hair or fur.   Fur models are effectively skinned hair, and are displayed with animation.

This document describes how to operate the viewer as well as how to bring new content into the viewer.

Contents

Basic Viewer Usage

Mouse and Keyboard

Dragging with the left mouse button turns the camera around the bear, i.e. it does not affect simulation or lighting.

Holding down the middle mouse button translates the bear, and the hair responds to the motion.

Holding down the control key while dragging with the middle mouse button rotates the bear rather than the camera.

Hitting ‘o’ brings up a file menu, to which you can browse to other .tfxproj files to load other models.

Rendering Parameters

Most of the rendering parameters are self-explanatory. I will focus on some of the less obvious ones.

Self-shadowing applies to the hair (shadow of hairs falling on other hairs). The number value scales the strength of the affect.

Density reduces the number of hairs that are rendered. The number simulated does not change.

Distance adaptive LOD shows a very low LOD version of the hair. A lower LOD is created by reducing the number of hairs while increasing their size. Parameters such as self-shadowing can also change to help keep the look more consistent.

Strand Copies simply duplicates each strand, slightly offset from the original. It is only duplicating for rendering, not simulation, so is sort of the opposite of density. It’s a very simple method, so is more useful to quickly test performance and look with more hair, and isn’t necessarily intended to be an authoring solution.

Thin Tip Causes the thickness of the hair to taper to a point. Turned off, the hair maintains a constant length through the strand.

Physics Parameters

Here are brief descriptions of the physics parameters. The best way to get a feel for what they do is to simply play with them. Some parameters are particular to a section of hair. There is a dropdown menu to select the current section.

Damping (per-section) smooths out motion of the hair. It also slows down the hair movement, making it more like hair under water, for example.

Local and Global Stiffness (per-section) makes each strand stiffer – meaning that if the hair started out straight, it will try to keep that shape. If it was curved, it will try to keep the curve. The difference between the two is that **global stiffness** tries to move the hair back to where it was originally in a global way – so if there is something pushing a straight hair in the middle, the tip will try to go back around the ball to where it started, giving it a curved shape. The **local stiffness** only looks at the shape locally, so if that same straight hair is pushed in the middle, the hair will get out of the way of the ball, but try to stay straight, instead of curving around it.

Global Shape Range (per-section) controls how much of each hair strand is affected by the global shape stiffness. If the value is 0, then the stiffness is off (as if it also has value of 0). If it is 1, then it affects the whole hair strand, and the tip of the hair will try to get back to where it was as described above. If the value is 0.5, then only the first half of the hair strand (from the root) tries to get back to its original position, so the tip will not try to get back to its original position in the previous example.

Length Constraint Iterations (global setting) allocates more simulation time to keeping the hair the right length. Try to keep this number as low as possible.

Local Shape Matching (global setting) allocates more simulation time to keeping the hair shape. It can make the hair seem stiffer, but also greatly increases simulation time.

Wind Magnitude (global setting) allows you to see the effect of wind on the hair.

Loading And Saving Different Models

The viewer reads a .tfxproj file describing the hair and the model the hair is attached to. By default, the viewer will load the animated bear.

You can also save changes you've made in the viewer. Most settings are saved in the .tfxproj file, which is a simple text file that may also be hand-edited. The button to save a tfxproj file is under the rendering panel.

Physics parameters tied to particular sections of hair are stored in a .tfxsim file, which is also a simple text file. The button for this is in the simulation menu. The default name for saving is the one already referred to by the tfxproj file. If you save under a different name, that new name will be recorded when you next save a tfxproj file.

One important limitation is that when saving, the tfxproj and tfxsim files must stay in the same directory, and that directory must be the same as the original tfxproj directory. So if you want to start a new directory, you must start by copying the tfxproj and tfxsim files to that new directory.

The viewer is not meant as a long term solution for TressFX parameter authoring, but is being released with some of that functionality in this release.

New Hair Content

In this section, we describe how to generate new hair content and see it in the viewer using a simple example. This requires three parts, which correspond to three files, and a fourth file that ties them together. There is also an optional hair color texture, which is a fifth file. The "Viewer Only" annotation relates parts that are specific to the viewer, and would likely be managed through a specific engine's system.

1) Mesh (.sdkmesh – Viewer Only)
2) Hair Geometry (.tfx)
3) Physics Parameters (.tfxsim)
4) Project File (.tfxproj – Viewer Only)
5) Hair Color Texture (.DDS – Viewer Only)

Hair Geometry (.tfx)

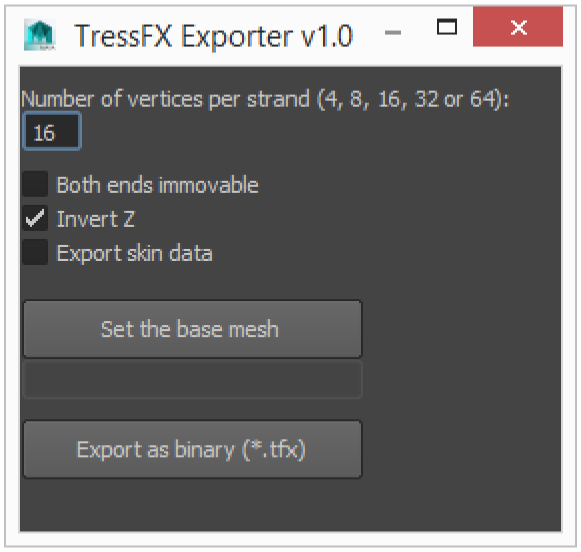
TressFX is designed to be compatible with your favorite hair modeler. All modeling is done within the modeling system of your choice, as long as you can turn them into splines at the end. We've used Shave and a Haircut, XGen and the 3ds Max native hair modeler, although the 3dS Max plugin is not included in this release.

In this section, we describe how to export your hair geometry as a .tfx file through our Maya exporter, beginning with instructions on how to install the plugin.

Installing the Plugin

The Maya TressFX exporter is a single python file located in the following location: AMD_TressFX_Tools\MayaPlugin\src \TressFX_Exporter.py. To enable this plugin, follow these steps.

1. Copy TressFX_Exporter.py into Maya's python script folder such as C:\Users\USER_NAME\Documents\maya\scripts
2. Launch Maya.
3. Open Script Editor and run the following lines in Python tab.
```
import TressFX_Exporter
reload(TressFX_Exporter)
TressFX_Exporter.UI()
```
4. This will bring up the TressFX Exporter window as below.



TressFX Exporter v1.0

Number of vertices per strand (4, 8, 16, 32 or 64): 16

Both ends immovable
[✓ Invert Z]
Export skin data

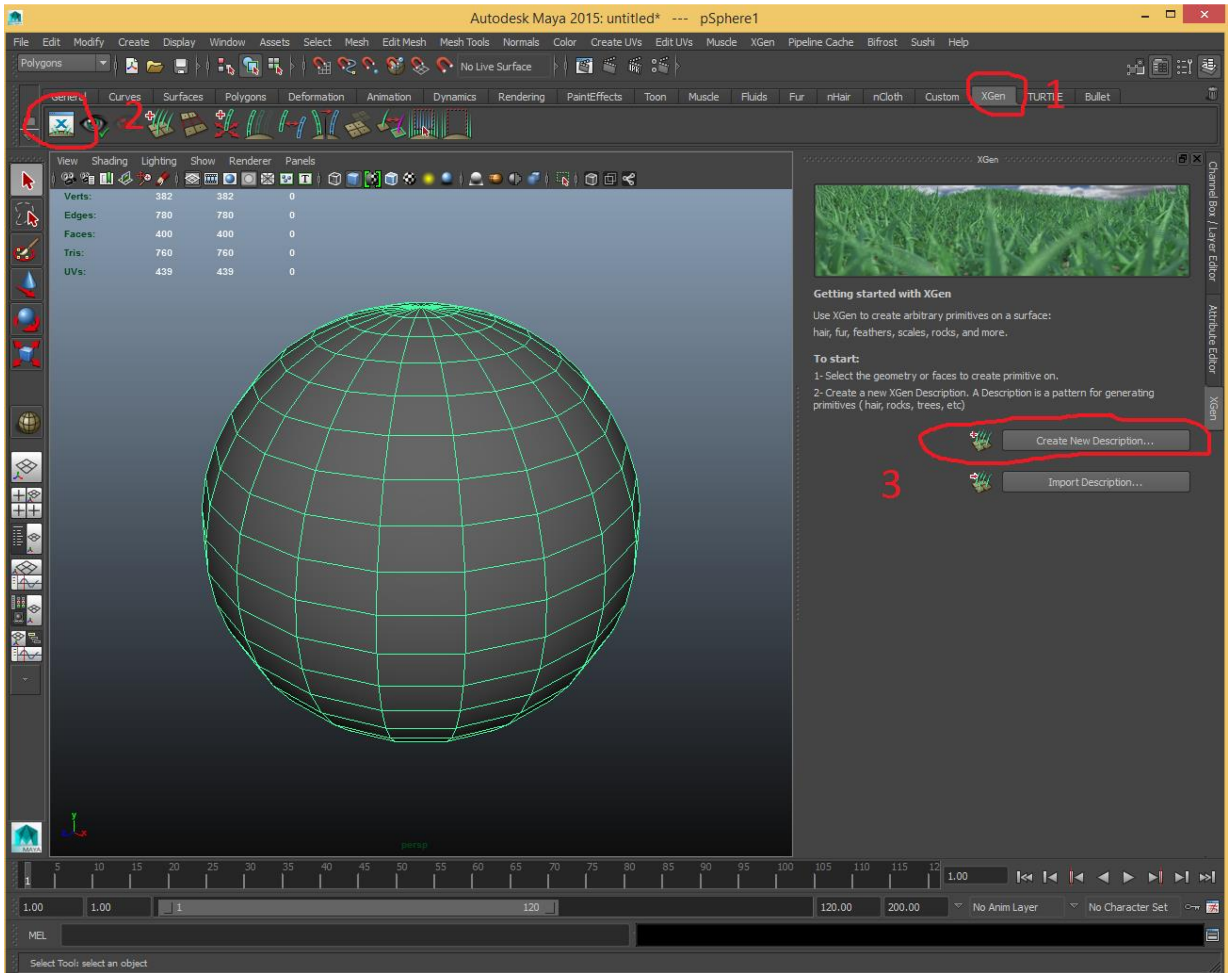
5. (Optional) The launch code can be shelved into Maya's toolbar so that user can launch it by clicking the toolbar button.

Exporting

As mentioned, you are free to use any hair modeling tool that creates nurbs curves (splines). In this walkthrough, we will be using XGen (specifically Maya 2015 version), which started shipping with Maya 2014, but requires a couple extra steps to convert to nurb curves.

Create a sphere (polygonal) with a radius of 100 and 20 subdivisions in each dimension.

Next, click on the XGen tab (1), and select the XGen window button on the left (2). You should see something like this:


Autodesk Maya 2015: untitled* --- pSphere1
File Edit Modify Create Display Window Assets Select Mesh Edit Mesh Mesh Tools Normals Color Create UVs Edit UVs Muscle XGen Pipeline Cache Bifrost Sushi Help
Polygons
No Live Surface
General Curves Surfaces Polygons Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur nHair nCloth Custom XGen TURTLE Bullet
1
2
View Shading Lighting Show Renderer Panels
Verts: 382 382 0
Edges: 780 780 0
Faces: 400 400 0
Tris: 760 760 0
UVs: 439 439 0
XGen
Create New Description...
Import Description...
3
1.00
120.00
200.00
No Anim Layer
No Character Set
MEL
Select Tool: select an object

Getting started with XGen

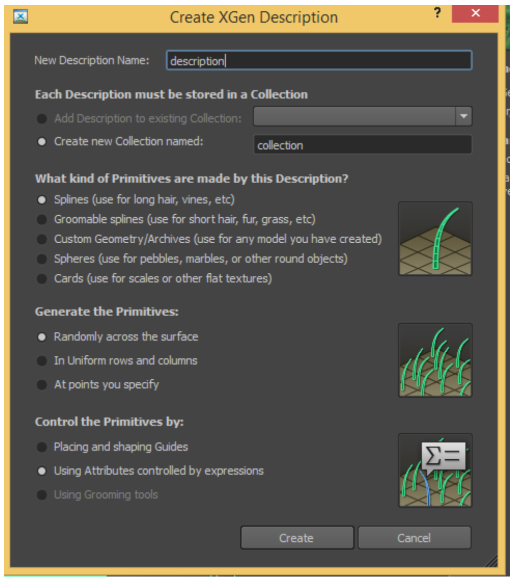
Use XGen to create arbitrary primitives on a surface: hair, fur, feathers, scales, rocks, and more.

To start:

1- Select the geometry or faces to create primitive on.

2- Create a new XGen Description. A Description is a pattern for generating primitives (hair, rocks, trees, etc)

Select the sphere and click on the “Create New Description…” (3) which should bring up this:



Create XGen Description

New Description Name: description

Each Description must be stored in a Collection

- Add Description to existing Collection:
- [✓ Create new Collection named:] collection

What kind of Primitives are made by this Description?

- [✓ Splines (use for long hair, vines, etc)]
- Groomable splines (use for short hair, fur, grass, etc)
- Custom Geometry/Archives (use for any model you have created)
- Spheres (use for pebbles, marbles, or other round objects)
- Cards (use for scales or other flat textures)

Generate the Primitives:

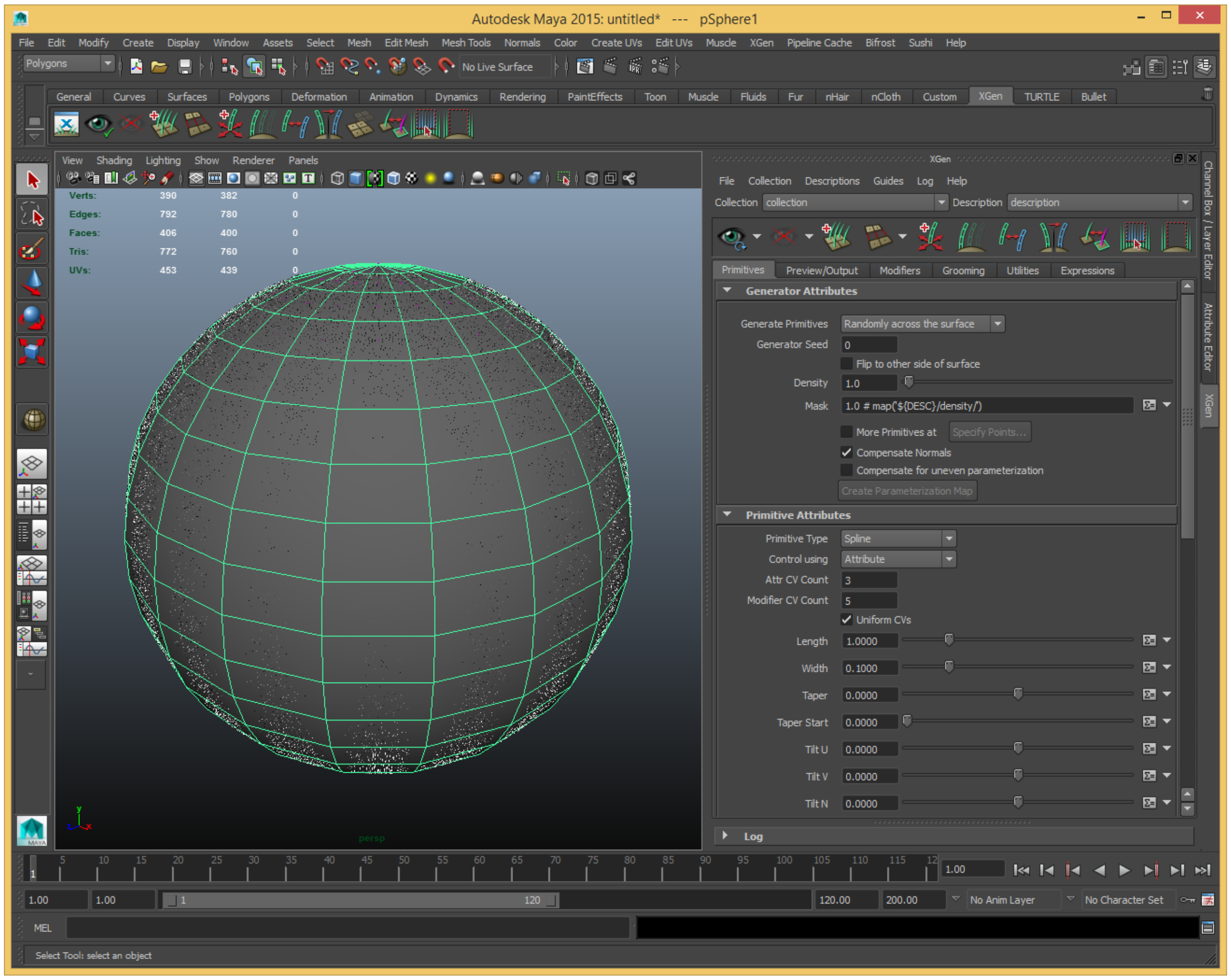
- [✓ Randomly across the surface]
- In Uniform rows and columns
- At points you specify

Control the Primitives by:

- Placing and shaping Guides
- [✓ Using Attributes controlled by expressions]
- Using Grooming tools

Create | Cancel

And choose “create”, which should give you something like this:



Autodesk Maya 2015: untitled* --- pSphere1
File Edit Modify Create Display Window Assets Select Mesh Edit Mesh Mesh Tools Normals Color Create UVs Edit UVs Muscle XGen Pipeline Cache Bifrost Sushi Help
Polygons
No Live Surface
General Curves Surfaces Polygons Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur nHair nCloth Custom XGen TURTLE Bullet
View Shading Lighting Show Renderer Panels
persp
XGen
File Collection Descriptions Guides Log Help
Primitives Preview/Output Modifiers Grooming Utilities Expressions
Log
No Anim Layer
No Character Set
MEL
Select Tool: select an object

Verts:	390	382	0
Edges:	792	780	0
Faces:	406	400	0
Tris:	772	760	0
UVs:	453	439	0

Collection: collection Description: description

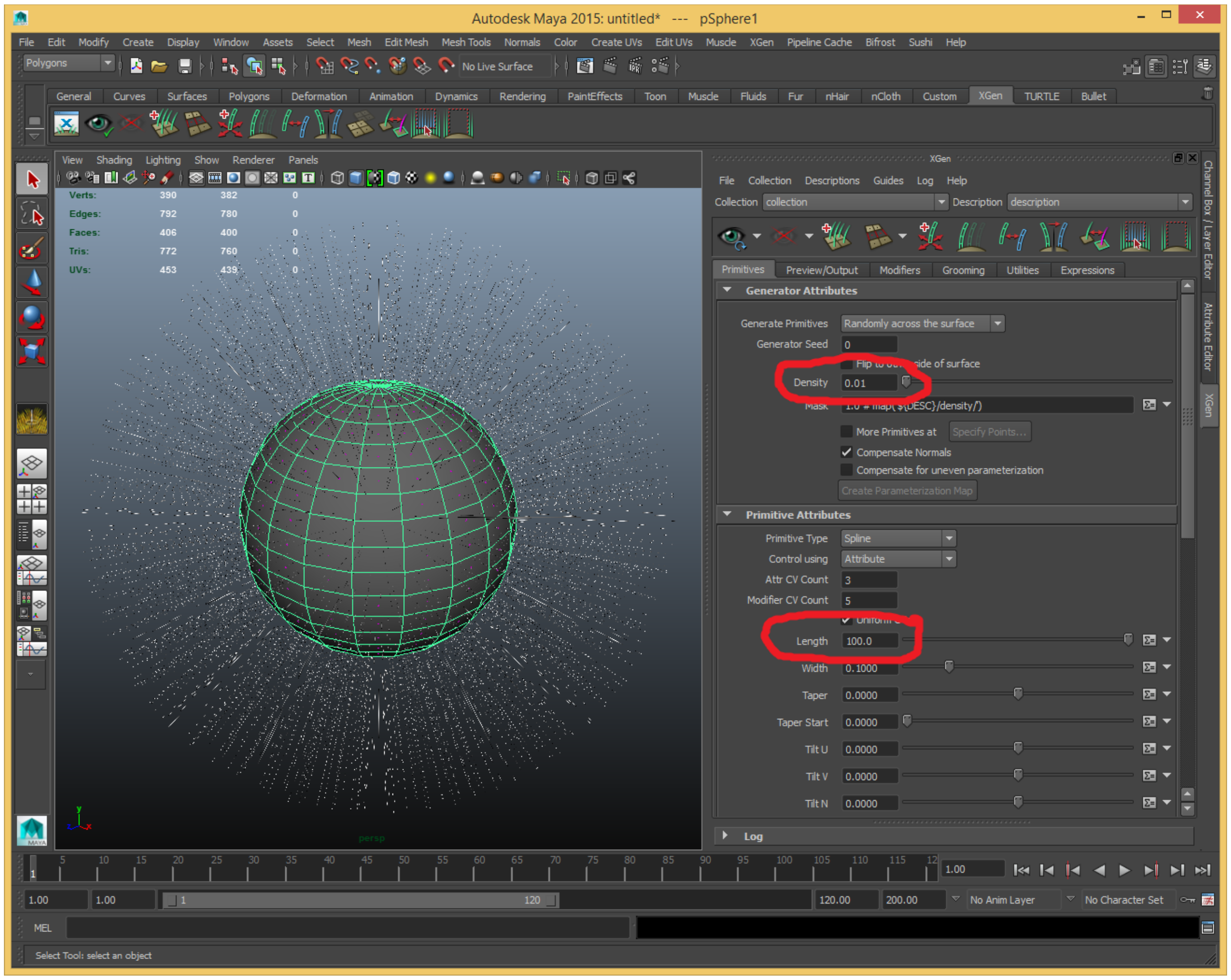
Generator Attributes

- Generate Primitives: Randomly across the surface
- Generator Seed: 0
- Flip to other side of surface
- Density: 1.0
- Mask: 1.0 # map('${DESC}/density/')
- More Primitives at: Specify Points...
- ✓ Compensate Normals
- Compensate for uneven parameterization
- Create Parameterization Map

Primitive Attributes

- Primitive Type: Spline
- Control using: Attribute
- Attr CV Count: 3
- Modifier CV Count: 5
- ✓ Uniform CVs
- Length: 1.0000
- Width: 0.1000
- Taper: 0.0000
- Taper Start: 0.0000
- Tilt U: 0.0000
- Tilt V: 0.0000
- Tilt N: 0.0000

Next, set the density(1) to 0.01 and the length(2) to 10, and you should have the following. Note that the line aliasing you see in the image is also in Maya because the geometry is quite thin – you can increase width to eliminate this affect, although the width is not exported at this time, as it is set through other means.



Autodesk Maya 2015: untitled* --- pSphere1
File Edit Modify Create Display Window Assets Select Mesh Edit Mesh Mesh Tools Normals Color Create UVs Edit UVs Muscle XGen Pipeline Cache Bifrost Sushi Help
Polygons
No Live Surface
General Curves Surfaces Polygons Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur nHair nCloth Custom XGen TURTLE Bullet
View Shading Lighting Show Renderer Panels
XGen
File Collection Descriptions Guides Log Help
Primitives Preview/Output Modifiers Grooming Utilities Expressions
Generator Attributes
Primitive Attributes
Log
MEL
Select Tool: select an object

Verts:	390	382	0
Edges:	792	780	0
Faces:	406	400	0
Tris:	772	760	0
UVs:	453	439	0

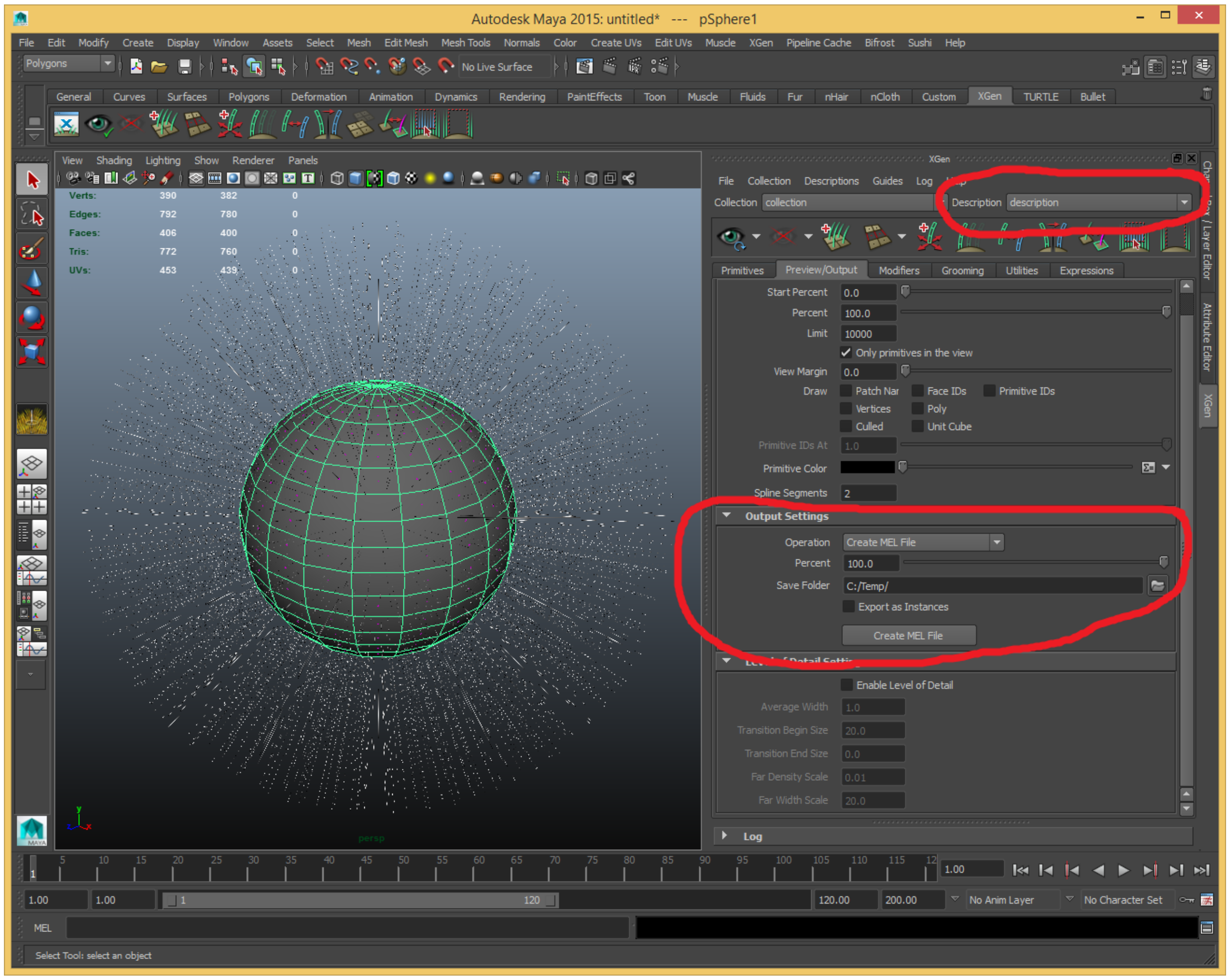
Collection: collection   Description: description

Generate Primitives: Randomly across the surface
Generator Seed: 0
Flip to outside of surface
Density: 0.01
Mask: 1.0 * map('${DESC}/density/')
More Primitives at: Specify Points...
✓ Compensate Normals
Compensate for uneven parameterization
Create Parameterization Map

Primitive Type: Spline
Control using: Attribute
Attr CV Count: 3
Modifier CV Count: 5
✓ Uniform CV
Length: 100.0
Width: 0.1000
Taper: 0.0000
Taper Start: 0.0000
Tilt U: 0.0000
Tilt V: 0.0000
Tilt N: 0.0000

Time slider: 1.00, 1.00, 1, 120, 120.00, 200.00, No Anim Layer, No Character Set

This is enough for illustration, so we'll move onto exporting. As mentioned before, we need to convert this to splines. In XGen, this is done by using the "Preview/Output" tab to export a mel file, as illustrated below.


Autodesk Maya 2015: untitled* --- pSphere1
File Edit Modify Create Display Window Assets Select Mesh Edit Mesh Mesh Tools Normals Color Create UVs Edit UVs Muscle XGen Pipeline Cache Bifrost Sushi Help
General Curves Surfaces Polygons Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur nHair nCloth Custom XGen TURTLE Bullet
View Shading Lighting Show Renderer Panels
XGen
File Collection Descriptions Guides Log
Primitives Preview/Output Modifiers Grooming Utilities Expressions
Output Settings
Create MEL File
Level of Detail Settings
Log
persp
No Anim Layer
No Character Set
MEL
Select Tool: select an object

Verts:	390	382	0
Edges:	792	780	0
Faces:	406	400	0
Tris:	772	760	0
UVs:	453	439	0

Collection: collection
Description: description

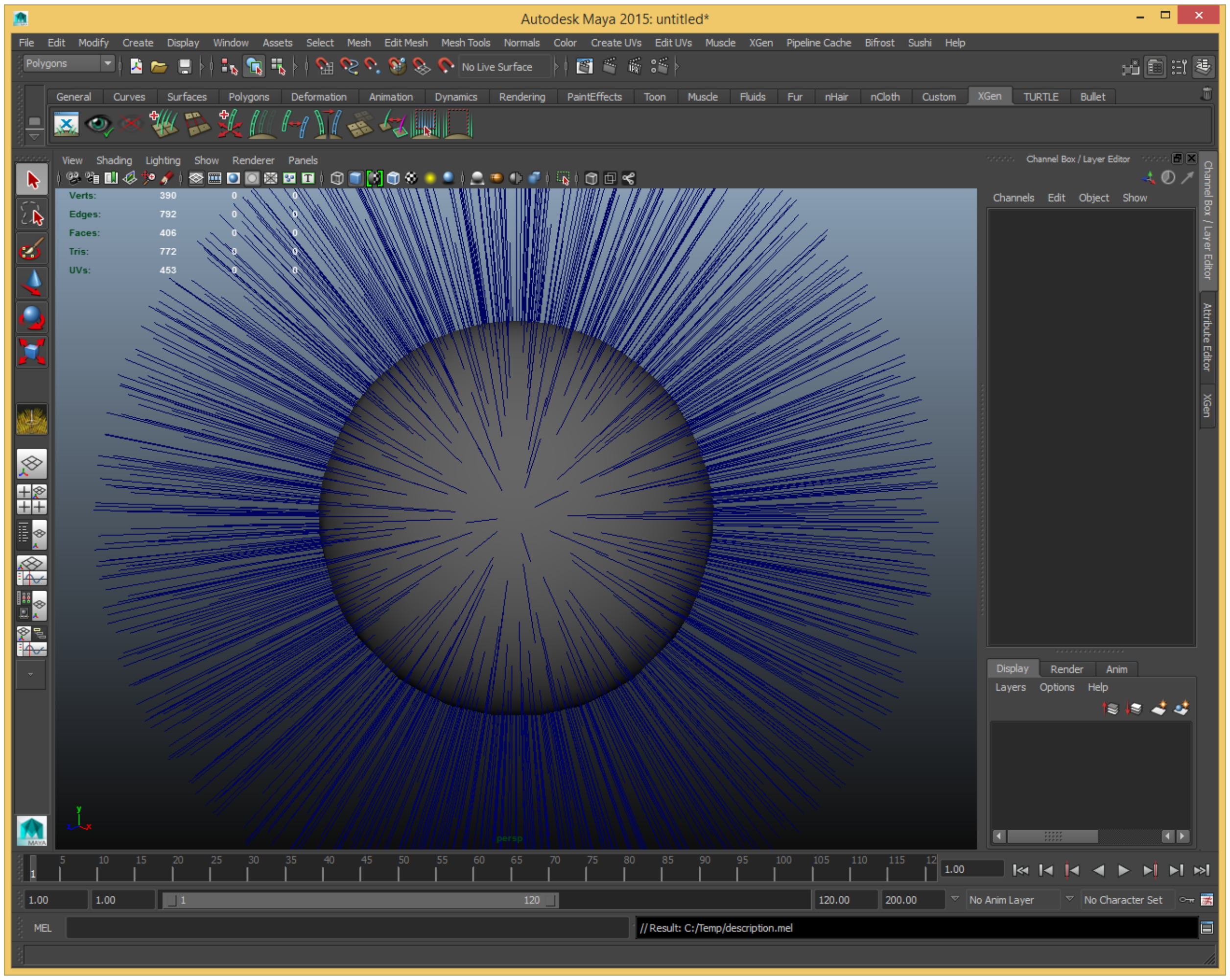
Start Percent: 0.0
Percent: 100.0
Limit: 10000
✓ Only primitives in the view
View Margin: 0.0
Draw: Patch Names, Face IDs, Primitive IDs, Vertices, Poly, Culled, Unit Cube
Primitive IDs At: 1.0
Primitive Color
Spline Segments: 2

Output Settings
Operation: Create MEL File
Percent: 100.0
Save Folder: C:/Temp/
Export as Instances

Enable Level of Detail
Average Width: 1.0
Transition Begin Size: 20.0
Transition End Size: 0.0
Far Density Scale: 0.01
Far Width Scale: 20.0

This will write the mel file the the specified directory. The name will depend on the name of the XGen "description".

If you like, you can hide or delete the XGen "collection" in the outliner, and then import the mel file. You should see something like this now:

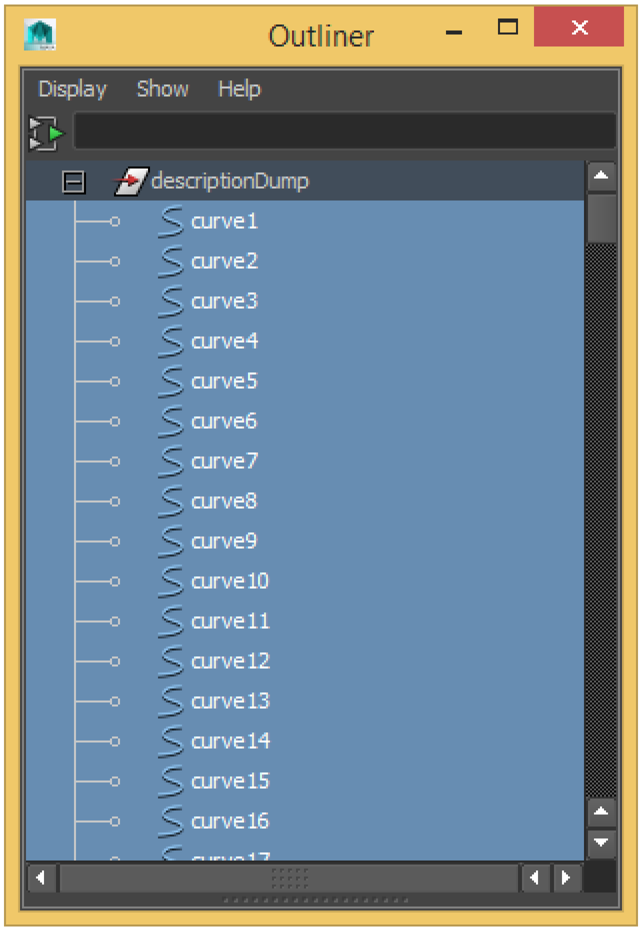

Autodesk Maya 2015: untitled*
File Edit Modify Create Display Window Assets Select Mesh Edit Mesh Mesh Tools Normals Color Create UVs Edit UVs Muscle XGen Pipeline Cache Bifrost Sushi Help
General Curves Surfaces Polygons Deformation Animation Dynamics Rendering PaintEffects Toon Muscle Fluids Fur nHair nCloth Custom XGen TURTLE Bullet
View Shading Lighting Show Renderer Panels
Channel Box / Layer Editor
Channels Edit Object Show
Display Render Anim
Layers Options Help
No Anim Layer
No Character Set
MEL

Verts:	390	0	0
Edges:	792	0	0
Faces:	406	0	0
Tris:	772	0	0
UVs:	453	0	0

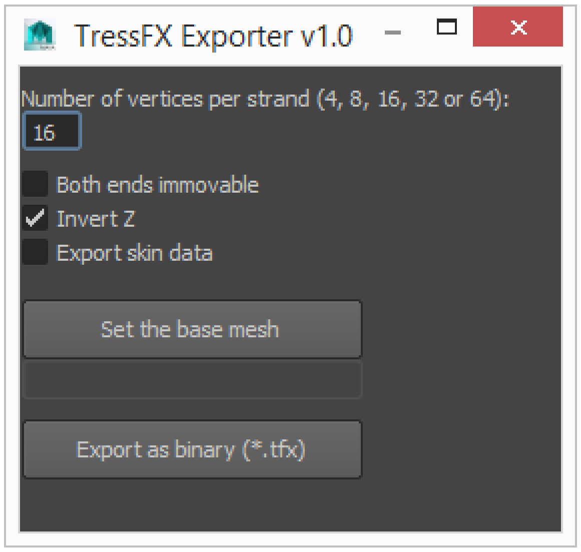
```
// Result: C:/Temp/description.mel
```

This now shows the nurb curves we wish to export.

To export, select the curves. It’s probably easiest to do this in the outliner. You need to select the actual curve nodes, not a node above them. *This will be fixed in a future release.*



The viewer uses a different coordinate system than Maya. Therefore, you will need to select the "invert Z" selection in the TressFX exporter, which is shown here again for convenience.


TressFX Exporter v1.0

Number of vertices per strand (4, 8, 16, 32 or 64): 16

- [ ] Both ends immovable
- [x] Invert Z
- [ ] Export skin data

Then you simply click the 'Export as binary(*.tfx)' button, which should bring up a file dialog. Once you hit save, you should see a progress bar. For this example, we'll assume you called it "sphere.tfx".

We need to copy this to a folder the viewer can access. For this example, create the subfolder AMD_TressFX_Viewer\media\sphere, and put it there.

Physics Parameters (.tfxsim)

This file contains physics parameters for the hair. You can copy an existing one to start from

```
AMD_TressFX_Viewer\media\testhair1\TestHair1.tfxsim
```

To

```
AMD_TressFX_Viewer\media\sphere\sphere.tfxsim
```

Visual Mesh (.sdkmesh – viewer only)

Now we will add the mesh to the scene. To do so, we export the mesh as an FBX® file as shown. We are not exporting an animation.

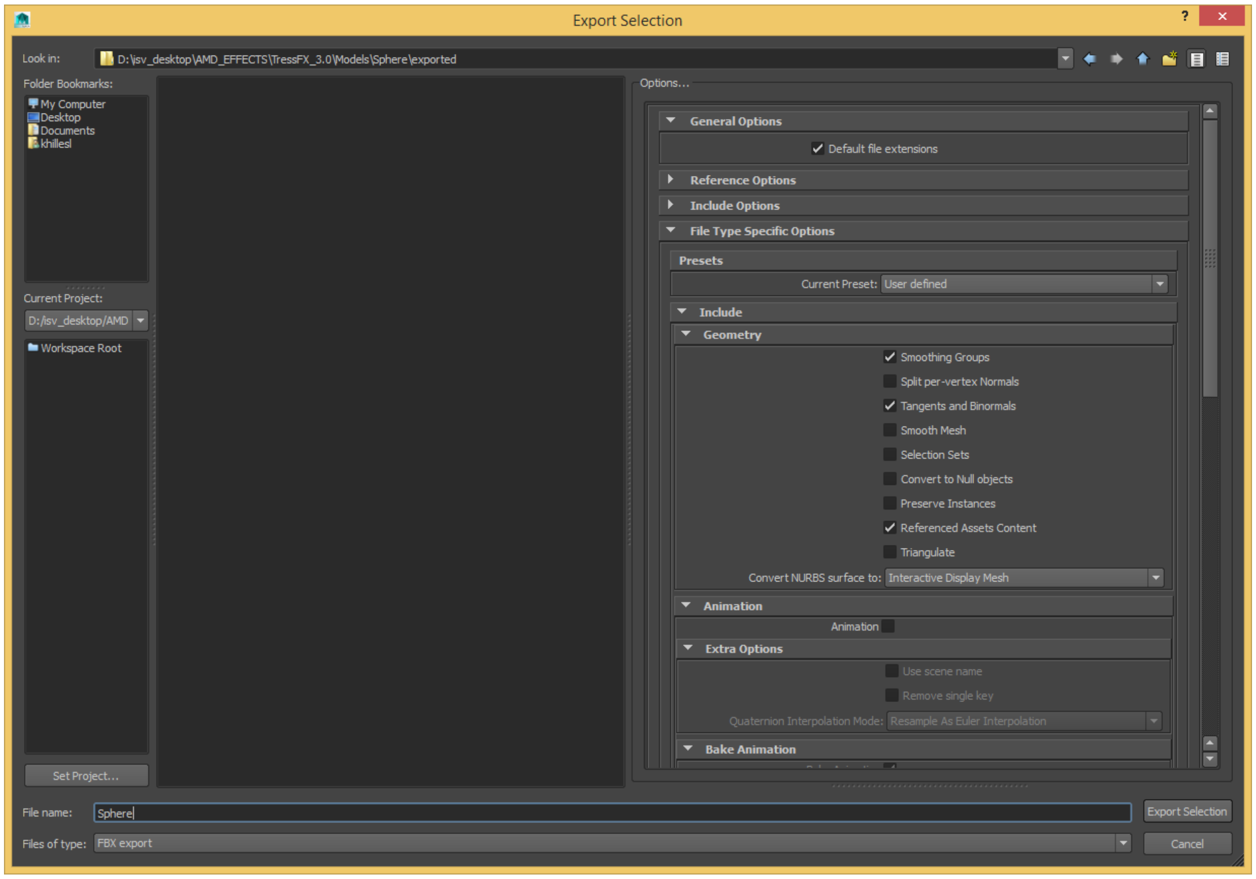
The mesh needs a Maya material, such as lambert, and a texture (file) in the color slot. The Ambient color setting is also observed. No other material slots are supported at this time. For this demo, I've attached a "gradient" texture to the color slot.

This will be a twostep process. First, we will export an FBX file. Then we will process that FBX file for use in the viewer.

Create a directory and subdirectory here:

```
AMD_TressFX_Example_Models\sphere
AMD_TressFX_Example_Models\sphere\exported
```

Export the mesh as an FBX file to `AMD_TressFX_Example_Models\sphere\exported`.


Export Selection

Look in: D:\isv_desktop\AMD_EFFECTS\TressFX_3.0\Models\Sphere\exported

Current Project: D:/isv_desktop/AMD

General Options
- [✓] Default file extensions

File Type Specific Options

Presets
- Current Preset: User defined

Include
Geometry
- [✓] Smoothing Groups
- [ ] Split per-vertex Normals
- [✓] Tangents and Binormals
- [ ] Smooth Mesh
- [ ] Selection Sets
- [ ] Convert to Null objects
- [ ] Preserve Instances
- [✓] Referenced Assets Content
- [ ] Triangulate
- Convert NURBS surface to: Interactive Display Mesh

Animation
- Animation [ ]

Extra Options
- [ ] Use scene name
- [ ] Remove single key
- Quaternion Interpolation Mode: Resample As Euler Interpolation

File name: Sphere

Files of type: FBX export

This file must then be turned into an sdkmesh file using ContentExporter. This is a tool provided by Microsoft. We have included a built copy for convenience in

```
 AMD_TressFX_Example_Models\ContentExporter\ContentExporter.exe
```

We have a batch file that invokes the tool with the correct arguments if run from the right relative directory. It needs to be run from the same directory as the FBX file. Again, we are assuming you've saved Sphere.fbx to AMD_TressFX_Example_Models\sphere\exported.

From there, you can run

```
..\..\ContentExporter\ProcessStaticFBX.bat sphere.fbx
```

This produces Sphere.sdkmesh in the directory, along with any referenced textures as .DDS files. These files need to be copied to

```
        AMD_TressFX_Viewer\media\sphere\
```

Project file (.tfxproj)

Next, we will create a tfxproj file. Use a text editor to create sphere.tfxproj in the same directory as the tfx file from the previous step, i.e., something like AMD_TressFX_Viewer\Media\Sphere\Sphere.tfxproj. The project file only needs two lines.

```
        sphere.tfx 0 0.1 5 sphere.tfxsim sphere
```

```
sphere.sdkmesh
```

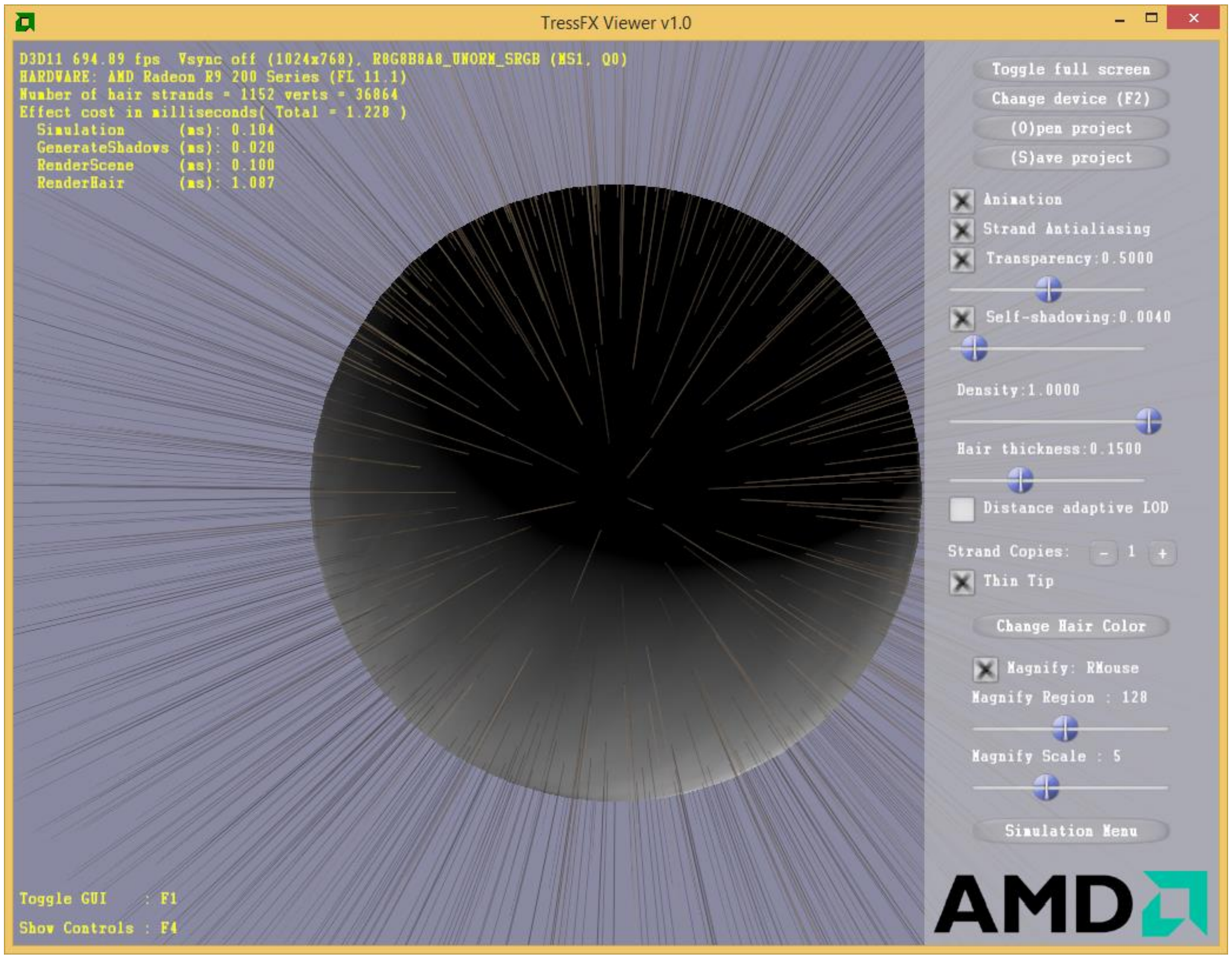
In the first line, the numbers are for runtime guide hair generation, which we are not using in this example (the zero turns it off). We also supply the name of the file containing physics parameters (sphere.tfxsim). The name at the end is the name of this part of the hair as seen in the viewer menu.

The second line simply gives the name of the mesh for the hair.

Now the contents of AMD_TressFX_Viewer\Media\sphere should be

```
sphere.tfx
sphere.sdkmesh
sphere.tfxsim
sphere.tfxproj
gradient.dds
```
(whatever DDS file was generated when created the sdkmesh file)

Running with the new model, you should now see the following (depending on texture)



```
D3D11 694.89 fps  Vsync off (1024x768), R8G8B8A8_UNORM_SRGB (MS1, Q0)
HARDWARE: AMD Radeon R9 200 Series (FL 11.1)
Number of hair strands = 1152 verts = 36864
Effect cost in milliseconds( Total = 1.228 )
  Simulation      (ms): 0.104
  GenerateShadows (ms): 0.020
  RenderScene     (ms): 0.100
  RenderHair      (ms): 1.087
```

Transparency:0.5000
Self-shadowing:0.0040
Density:1.0000
Hair thickness:0.1500
Strand Copies: 1
Magnify Region : 128
Magnify Scale : 5

Follow Hairs

The model is a bit sparse. Changing the following line in sphere.fxproj adds additional hairs with little additional simulation cost. The first number adds an additional 10x hair strands, clustered around each of the originals in a roughly 10 unit radius (second number) around the root of the original, and the tips in a 10 x 5 unit radius (second number times third number).

```
    Sphere.tfx 10 10 5 Sphere2.tfxsim sphere
```

Collision Shapes

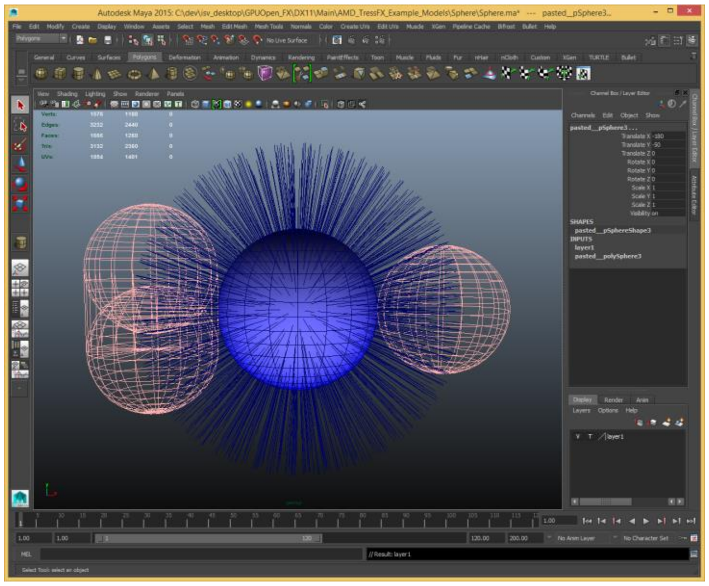
Collision shapes can be added so the hair does not go through the head or shoulders. For now, the viewer can only manage up to three collision spheres, and they are specified directly in the .tfxproj file.

The following two lines adds a sphere and a capsule collider.

```
    #                    x,  y,  z,  radius
    collision_sphere   180   0   0       80

    # capsule is two spheres and a cylinder sharing the same radius.
    # we give the center point of each sphere, and the shared radius.
    #                  first point        second    radius
    collision_capsule    -180 50 0    -180 -50 0        80
```

Here's how they look in Maya for the sake of visualization. We have not yet released a proper export system for this, so for now, you can use Maya to visualize where you want the shapes, and enter the appropriate data in the tfxproj file by hand.



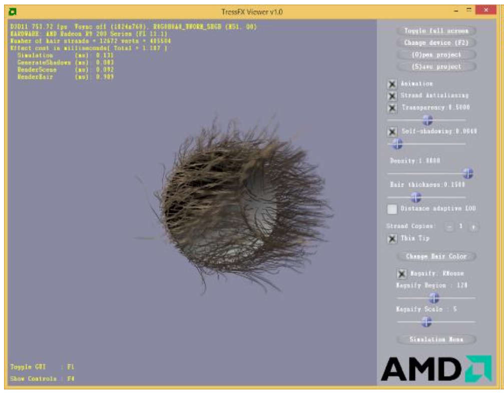
Note that the viewer uses a left-handed coordinate system with the z in the opposite direction of Maya, so to translate from Maya, you negate the z coordinate.

Final Tweaks

We change the sphere simulation parameters (.tfxsim) to make the hairs floppier.

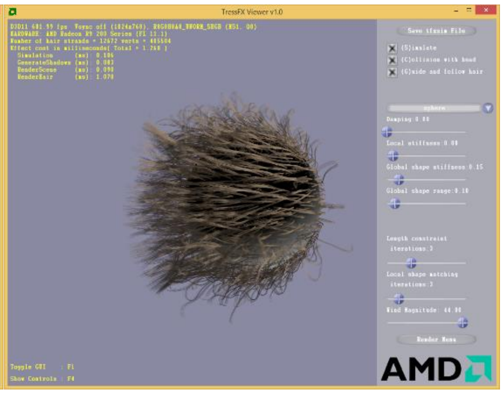
```
stiffnessForLocalShapeMatching 0.08
stiffnessForGlobalShapeMatching 0.15
globalShapeMatchingEffectiveRange 0.1
```

And then turn on wind to show how the hair forms around the collision objects, using both positive and negative magnitude to blow it in either direction.



```
D3D11 753.72 fps  Vsync off (1024x768), R8G8B8A8_UNORM_SRGB (MS1, Q0)
HARDWARE: AMD Radeon R9 200 Series (FL 11.1)
Number of hair strands = 12672 verts = 405504
Effect cost in milliseconds( Total = 1.187 )
  Simulation      (ms): 0.131
  GenerateShadows (ms): 0.083
  RenderScene     (ms): 0.092
  RenderHair      (ms): 0.909
```

- [x] Animation
- [x] Strand Antialiasing
- [x] Transparency 0.5000
- [x] Self-shadowing 0.0040
- Density 1.0000
- Hair thickness 0.1500
- [ ] Distance adaptive LOD
- Strand Copies: 1
- [x] Thin Tip
- [x] Magnify: RMouse
- Magnify Region : 128
- Magnify Scale : 5



```
D3D11 601.59 fps  Vsync off (1024x768), R8G8B8A8_UNORM_SRGB (MS1, Q0)
HARDWARE: AMD Radeon R9 200 Series (FL 11.1)
Number of hair strands = 12672 verts = 405504
Effect cost in milliseconds( Total = 1.268 )
  Simulation      (ms): 0.106
  GenerateShadows (ms): 0.083
  RenderScene     (ms): 0.090
  RenderHair      (ms): 1.070
```

- [x] (S)imulate
- [x] (C)ollision with head
- [x] (G)uide and follow hair
- Damping 0.00
- Local stiffness 0.08
- Global shape stiffness 0.15
- Global shape range 0.10
- Length constraint iterations 3
- Local shape matching iterations 3
- Wind Magnitude: 44.00

Scalp Coloring

You can get away with a little less thickness in the hair if you color the scalp under the hair a color that's close to the hair color (biased towards darker).

New Fur Content

Before reading this section, you should be aware of the process described for putting together a hair model. Much of the process is the same as hair, with a few additions. There are a total of six files now, with an optional seventh file.

1) Mesh and Rig (.sdkmesh – Viewer Only)
2) Animation (.sdkmesh_anim – Viewer Only)
3) Fur Geometry (.tfx)
4) Fur Skinning Information (.tfxsim)
5) Physics Parameters (.tfxsim)
6) Project File (.tfxproj)
7) Fur Texture File (optional)

We will be running through an example using the bear model. We will start with the bear maya file, but will re-export the bear model and hair for loading into the viewer. We'll make extra copies of files in the following directories, which you should create.

```
AMD_TressFX_Viewer\media\bear2
AMD_TressFX_Example_Models\bear2
AMD_TressFX_Example_Models\bear2\exported
```

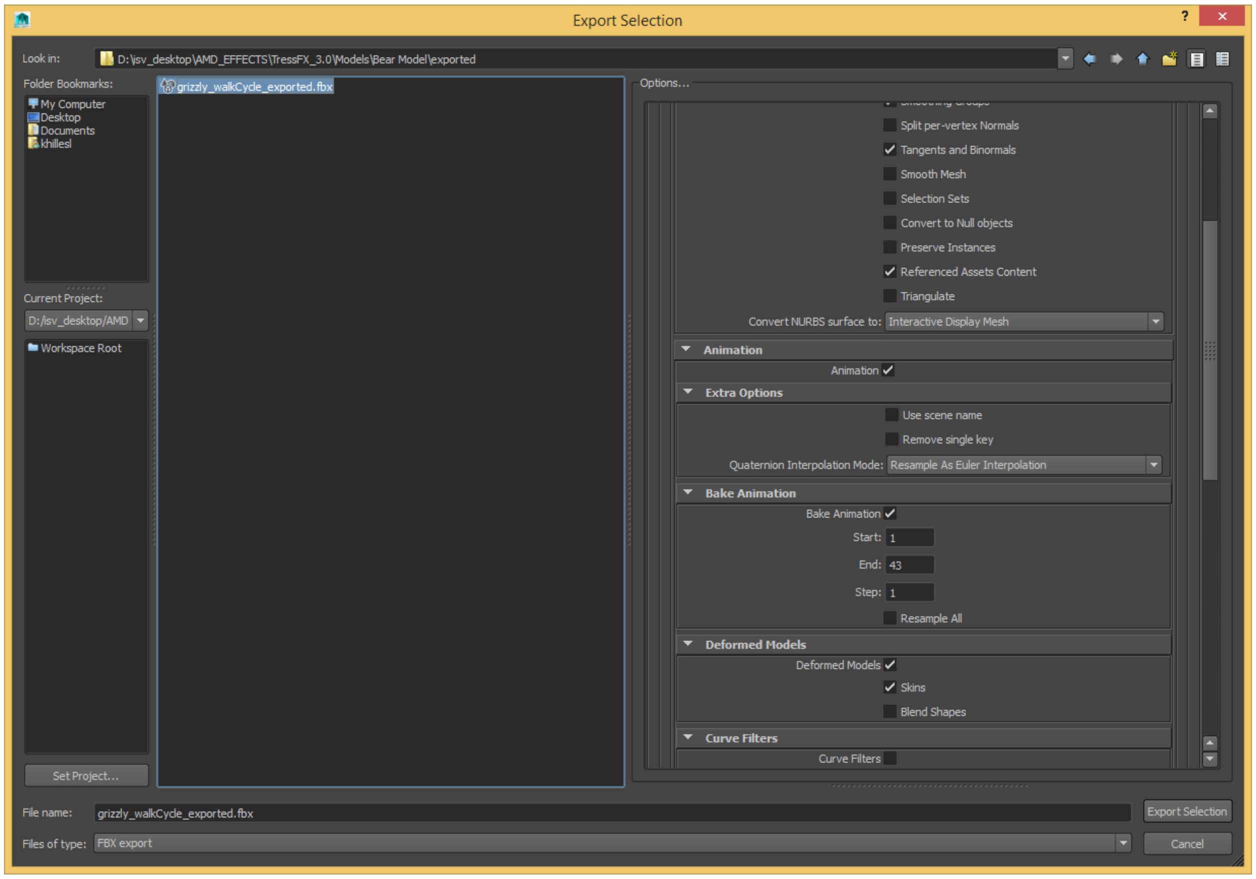
The bear Maya files are in AMD_TressFX_Example_Models, and are as follows:

t_Grizzly_weighted4.ma	Mesh and rig
grizzly_walkCycle.ma	Walk cycle animation. References t_Grizzly_weighted4.ma.
Grizzly_wFUR.rnd.ma	Fur file. References t_Grizzly_weighted4.ma.

Mesh and Rig (.sdkmesh) and Animation (.sdkmesh_anim)

The mesh, rig and animation are all exported from Maya to the FBX file. In this case, we load grizzly_walkCycle.ma.

Returning to the FBX export screen, we export the animation by including the animation checkbox. In this case, we are exporting the running animation along with the mesh and rig. Make sure you check the Animation and Deformed Models (Skins).



Look in: D:\isv_desktop\AMD_EFFECTS\TressFX_3.0\Models\Bear Model\exported

grizzly_walkCycle_exported.fbx

Current Project: D:/isv_desktop/AMD

- [ ] Split per-vertex Normals
- [✓] Tangents and Binormals
- [ ] Smooth Mesh
- [ ] Selection Sets
- [ ] Convert to Null objects
- [ ] Preserve Instances
- [✓] Referenced Assets Content
- [ ] Triangulate

Convert NURBS surface to: Interactive Display Mesh

Animation

Animation ✓

Extra Options

- [ ] Use scene name
- [ ] Remove single key

Quaternion Interpolation Mode: Resample As Euler Interpolation

Bake Animation

Bake Animation ✓

Start: 1

End: 43

Step: 1

- [ ] Resample All

Deformed Models

Deformed Models ✓

- [✓] Skins
- [ ] Blend Shapes

Curve Filters

Curve Filters

File name: grizzly_walkCycle_exported.fbx

Files of type: FBX export

Exporting these files into AMD_TressFX_Example_Models/bear2/exported as bear.fbx, we run the following from that directory.

```
..\..\ContentExporter\ProcessSkinnedFBX.bat bear.fbx
```

This creates both a bear2.sdkmesh, bear2.sdkmesh_anim, and 3 DDS files. These are copied into the following directory

```
AMD_TressFX_Viewer\media\bear2
```

The contents of that directory should now be:

```
bear_cm_ao.dds
bear_nm.dds
bear_sm.dds
bear.sdkmesh
bear.sdkmesh_anim
```

Fur Information (.tfx, .tfxsim, and .tfxskin)

Now we load the hair file: Grizzly_wFUR.rnd.ma

The fur was authored in XGen. Therefore, we need to convert it to nurb curves and export as before.

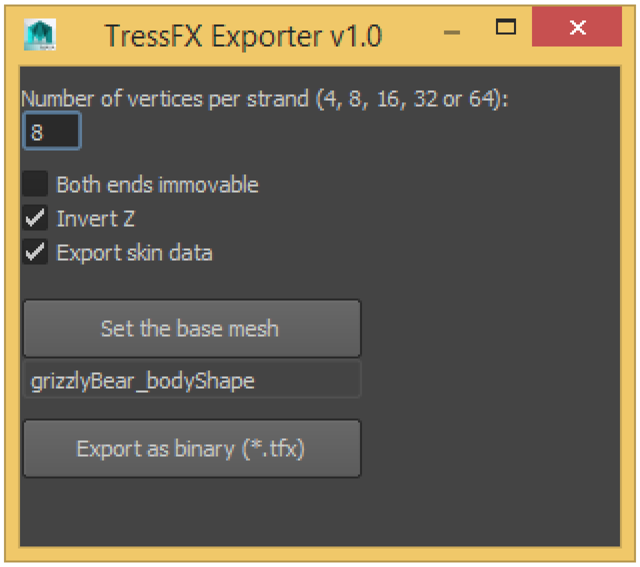
Once again, we select the XGen tab, and the XGen window on the left. The fur description is called "shag". We export the mel file. You should see something like this:

After exporting the mel file, we hide or delete the XGen "collection" in outliner, and import the mel file. You may need to turn on NURBS curve visualization to see them in the view panel (Show->NURBS curves).

Now the export process changes a bit for skinned models, because we need to extract some additional information from the mesh that the hair is attached to.

This time, we begin by selecting the base mesh – this is the mesh the hair will be attached to. In our case it is "grizzlyBear_body". In the TressFX exporter, push the "Set the base mesh" button. You should see "grizzlyBear_body" appear beneath it.

We make two other changes this time. First, because we have so many, shorter hairs, we will reduce the vertex count of each hair on export to 8, which is a setting in the exporter. We also check the "Export skin data" box. Here are the settings.



TressFX Exporter v1.0

Number of vertices per strand (4, 8, 16, 32 or 64):
8

- [ ] Both ends immovable
- [x] Invert Z
- [x] Export skin data

grizzlyBear_bodyShape

Once again, we select the NURBS curves in the outliner, and hit "Export as binary" again, saving the file as

```
AMD_TressFX_Viewer\media\bear2\bear.tfx
```

After that's done, you will be prompted to save a second file. This one should be saved as

```
AMD_TressFX_Viewer\media\bear2\bear.tfxskin
```

This second file contains information required for skinning, but also the UV coordinates of the hair for coloring the hair according to a texture at runtime. *This organization may changed in a future release.*

For a .tfxsim file, copy the existing bear one from

```
AMD_TressFX_Viewer\media\bear\bear.tfxsim
```

To

```
AMD_TressFX_Viewer\media\bear2\bear.tfxsim
```

Project File for Fur (.tfxproj)

A project file for fur is almost the same as a project file for hair, with two new files listed: the .sdkmesh_anim file for animation, and the .tfxskin file.   Here is the list of the minimum files required for a fur model.

```
.tfx
.tfxsim
.tfxskin
.sdkmesh
.sdkmesh_anim
.tfxproj
```

The project file includes references to all of these files. It essentially looks identical to a hair project file, but with the addition of the .sdkmesh_anim and .tfxskin files.

```
bear.tfx 0 0 5 bear.tfxsim 'all'

bear.tfxskin
bear.sdkmesh
bear.sdkmesh_anim
```

At this point, you should be able to load your copy of the bear in the viewer. It will look a bit different because it uses the default hair color settings of the viewer, which were designed for the default ponytail model.

Fur Shading Changes

To get the same appearance as the original bear, you need a few more changes. We use a texture to get variation in fur color, and tone down the specular term to reduce the shampoo-commerical-look.

First, copy the fur color texture to the new directory.

```
AMD_TressFX_Viewer\Media\Bear\fur_color.dds
```

To

```
AMD_TressFX_Viewer\Media\Bear2\fur_color.dds
```

Add the following lines to the .tfxproj file:

```
fur_color.dds
Ks1 0.01
Ks2 0.1
```

Ks1 and Ks2 are the specular coefficients on the hair (light-colored, and hair-colored respectively).

Viewer Skin Weights and Rig Limitations (Viewer only)

Only the four highest skinning weights are used for each vertex. Negative weights may do weird things. This is specific to the viewer.

The rig must not have more than 256 bones.

The viewer also assumes that the first mesh it loads is used to drive the hair. In other words, selecting the grizzlyBear_body for exporting hair works out ok, because it is also the first mesh in the list of meshes for the bear, and it only has one material. This will be fixed in future versions.

Fur is skinned by the first mesh in the viewer. This means that if multiple meshes are exported from Maya, the first mesh must be the one and only mesh with fur attached, and may not consist of more than one material. This will be fixed in future versions.

Caution on Maya File Referencing and Namespaces

The sdkmesh_anim file refers to the rig and mesh skinning information in sdkmesh using names from Maya. The easiest thing to do is export the one FBX file from the file with animation and create both sdkmesh and sdkmesh_anim from that.

You can theoretically apply different sdkmesh_anim files to the same sdkmesh as long as the names are consistent. A case where this would break down is if you took sdkmesh_anim that came from Grizzly_walkCycle.ma → FBX file → sdkmesh_anim, which references t_Grizzly_weighted4.ma, and tried that with t_Grizzly_weighted4.ma → FBX file → sdkmesh. That's because Grizzly_walkCycle.ma references t_Grizzly_weighted4.ma, adding a namespace that is not in t_Grizzly_weighted4.ma.

Additional tfxproj Settings

Here we see other settings possible in the tfxproj file, along with their defaults. You can get this list by saving the ponytail tfxproj file from the viewer, for example. For the most part, these are relatable to the names in the GUI, but we will define the hair shader constants as well.

```
lengthConstraintIterations 3
localShapeMatchingIterations 3
windMag 0

thickness 0.15
density 1
alpha 0.5
shadowMapAlpha 0.004

Kd 0.4                diffuse term multiplier.
Ks1 0.04              first spec multiplier. Color is from light.
Ex1 80                first spec exponent
Ks2 0.5               second spec multiplier. Color is from hair.
Ex2 8                 second spec exponent
hairColor 0.45098 0.329412 0.219608    hair color when not textured
```