UE4 TressFX 5.0

AMD GPU DevTech Engineer

Lisen Wang

Agenda

- UE4 TressFX 4.1 limitations
- UE4 TressFX 5.0
  - Simulation improvements
  - Rendering improvements
- Integration
- How to Use in UE4
- Future Work

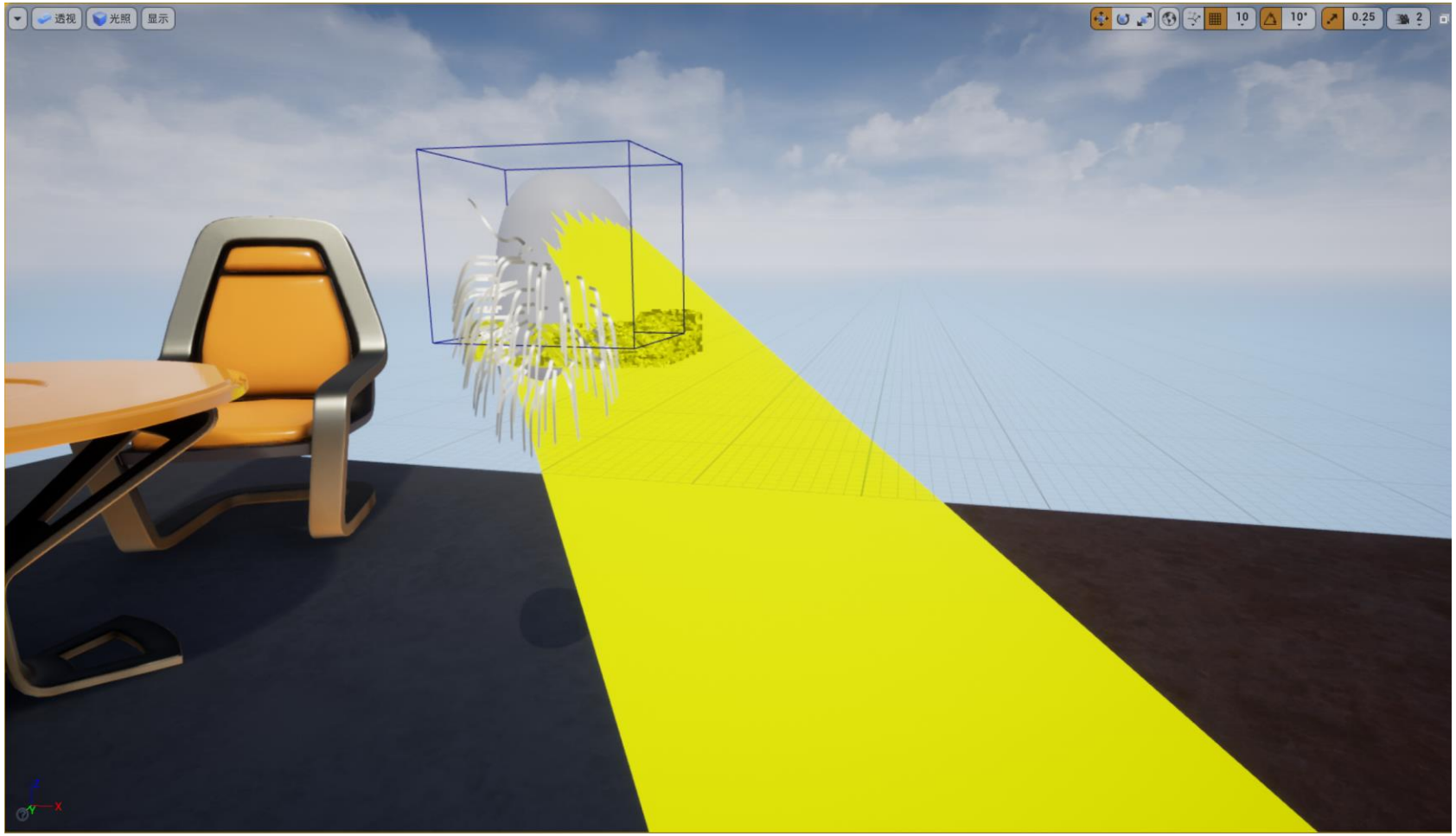
UE4 TRESSFX 4.1 LIMITATIONS

UE4 TressFX 4.1 limitations

- Maya Python Exporter
  - The number of binding bones limits to 4
  - Strands/Collision Mesh skinned twice if not export in the right way from Maya XGen
- Asset Editor & Visualization Tools
  - No support for LevelMap/Asset/Blueprint Editor well and lots of bugs
  - No support for Visualization tools to check whether asset is correct
- Simulation
  - Animation data used for simulation lags a frame behind UE4 Skinned Mesh animation data
  - Simulation Editor wasn't easy to use and no SDF Editor
  - SDF BoundingBox was computed on CPU side
  - SDF limitation which from developer feedback
- Rendering
  - No support for TAA
  - No support for SkyLight
  - No support for Cast/Receive Shadows
  - ResourceManagement was not reasonable enough which is very important to implement Editors

Limitations from developer feedback

- SDF Info is incorrect, yellow area is SDF Voxels



Limitations from developer feedback

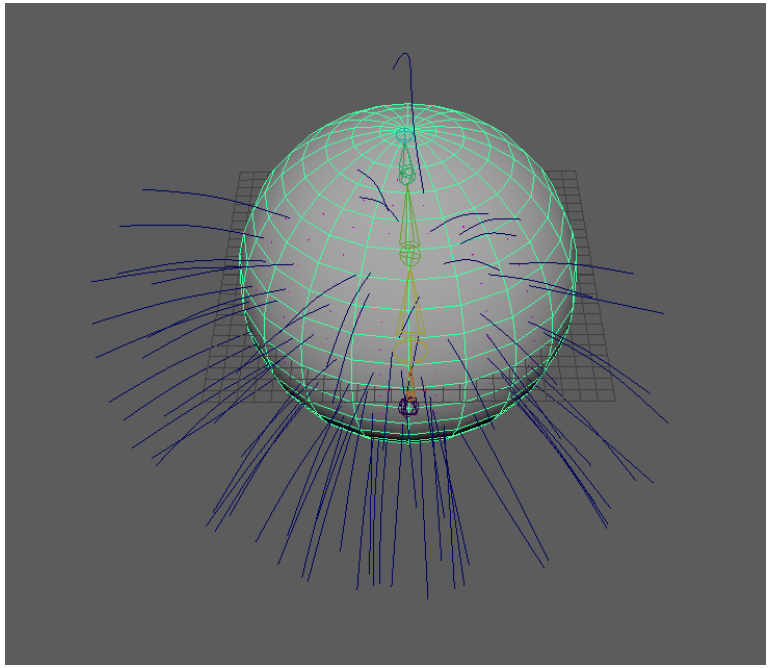
- Animation data used for simulation lags a frame behind UE4 Skinned Mesh animation data

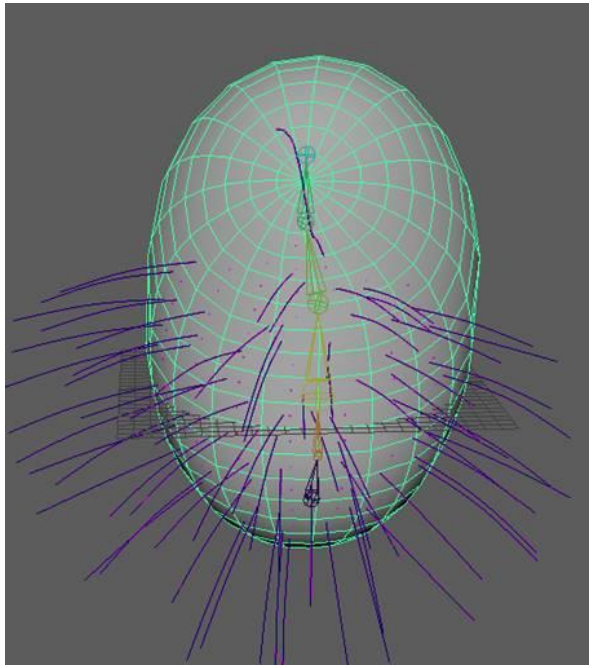
UE4 TRESSFX 5.0

How to implement improvements

- Prepare Simple Maya Asset for test
- Check whether the Asset is correct after it’s imported in UE4
- Validate the Collision Mesh Computation
- Reproduce simulation issue by making a video
- Validate the SDF Computation

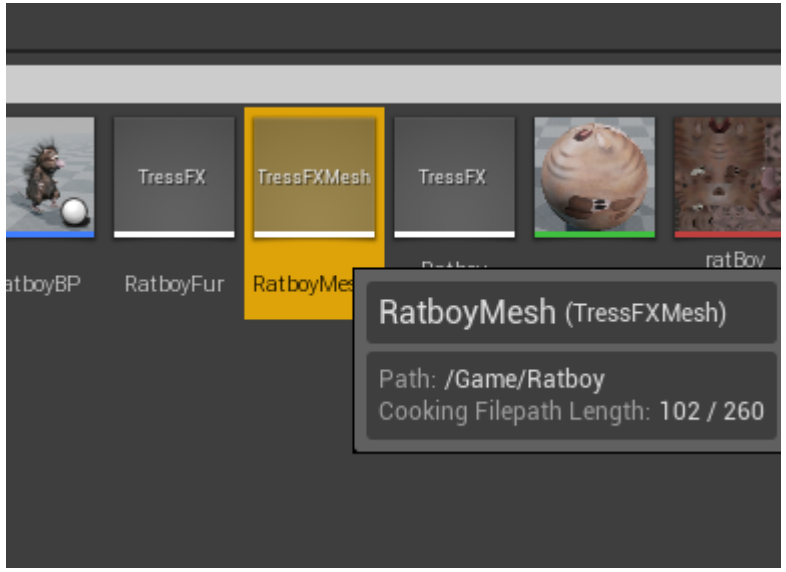
Simple Maya Asset for test

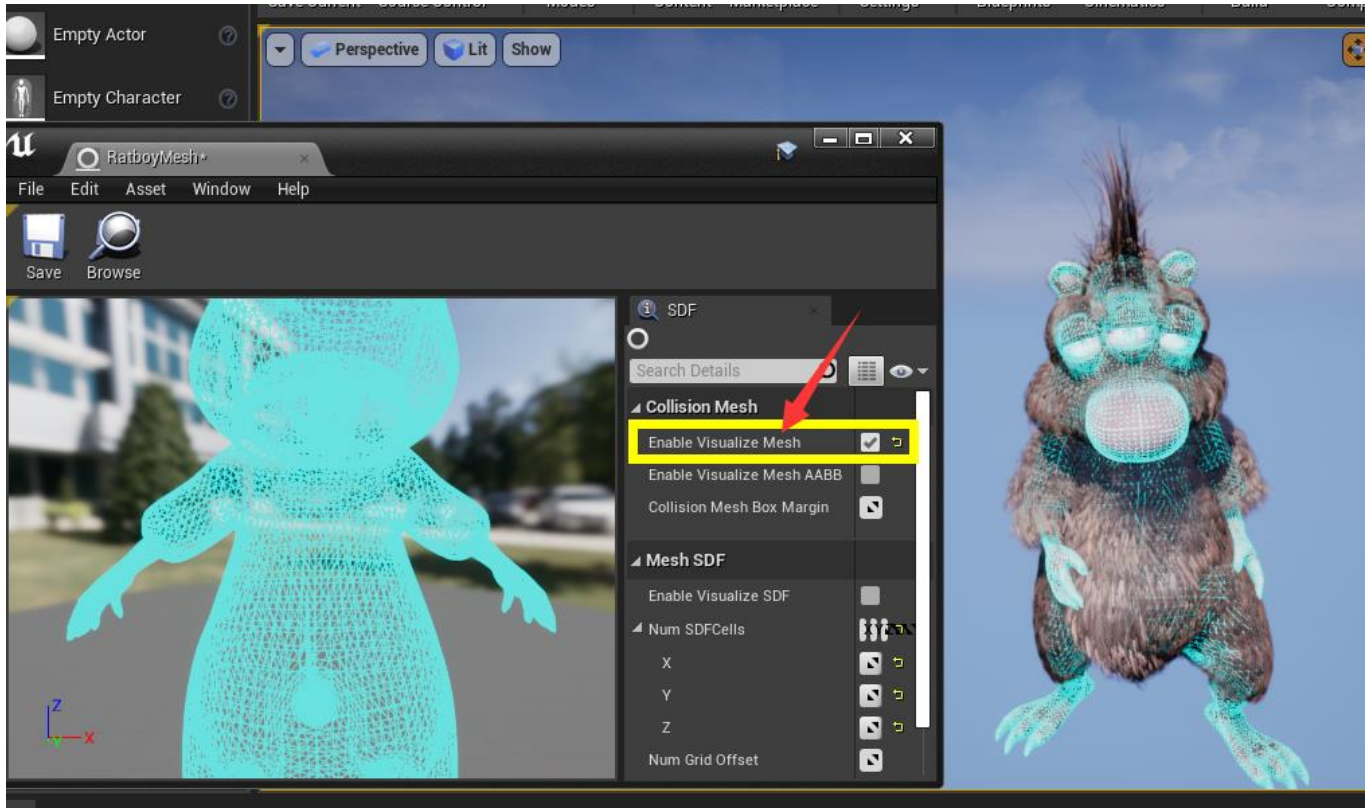




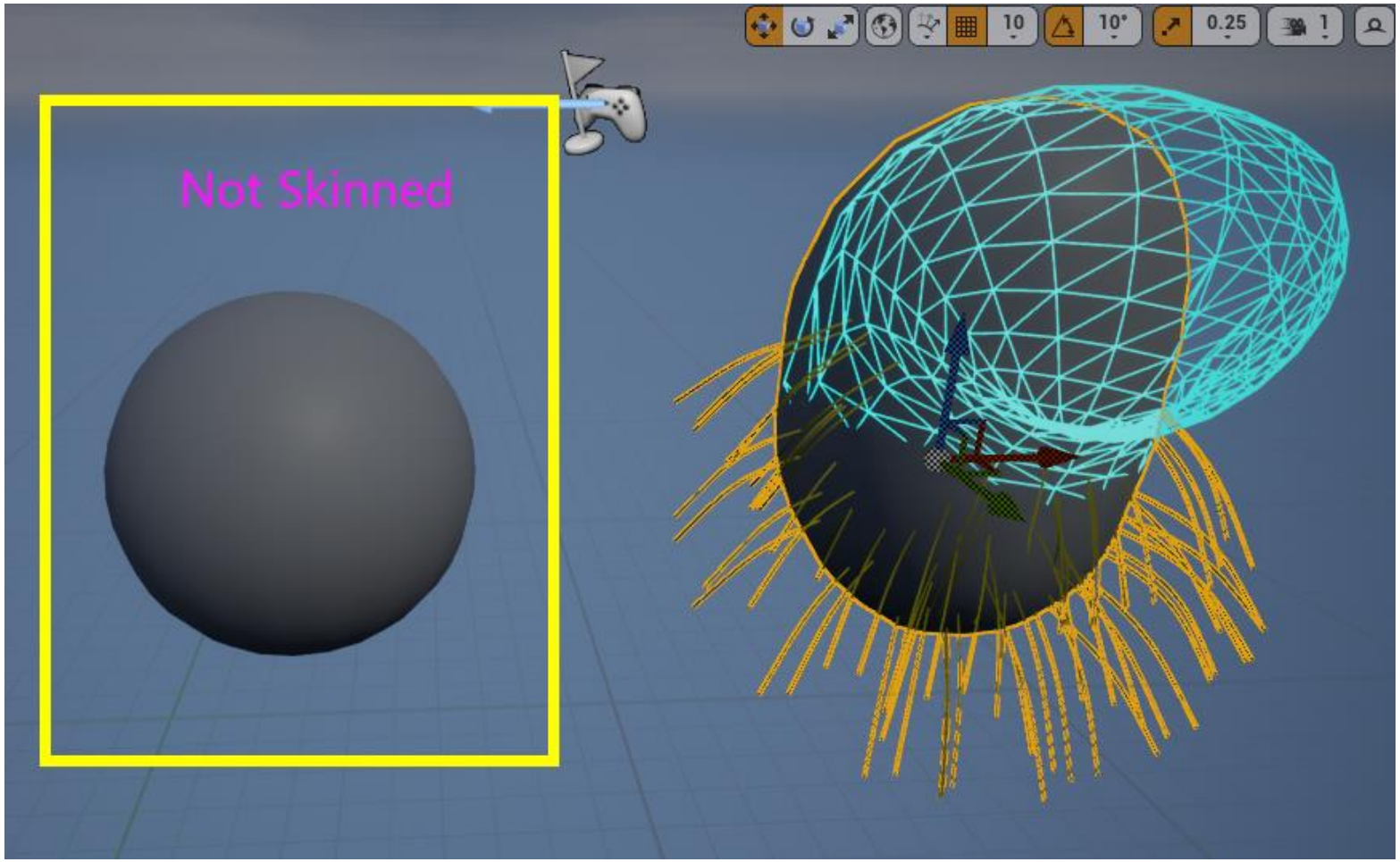
- Two Keyframes
- Check to export from each Keyframe
- Need to Visualize Collision Mesh in UE4

TressFX Collision Mesh





Import in UE4 to Validate the Collision Mesh Computation



- If export from the second Keyframe, it would result in collision mesh and strands skinned twice
- Should export from the right keyframe

Reproduce < Animation data used for simulation lags a frame behind UE4 Skinned Mesh animation data>

- If the animation plays very fast,

it’s not easy to reproduce this issue

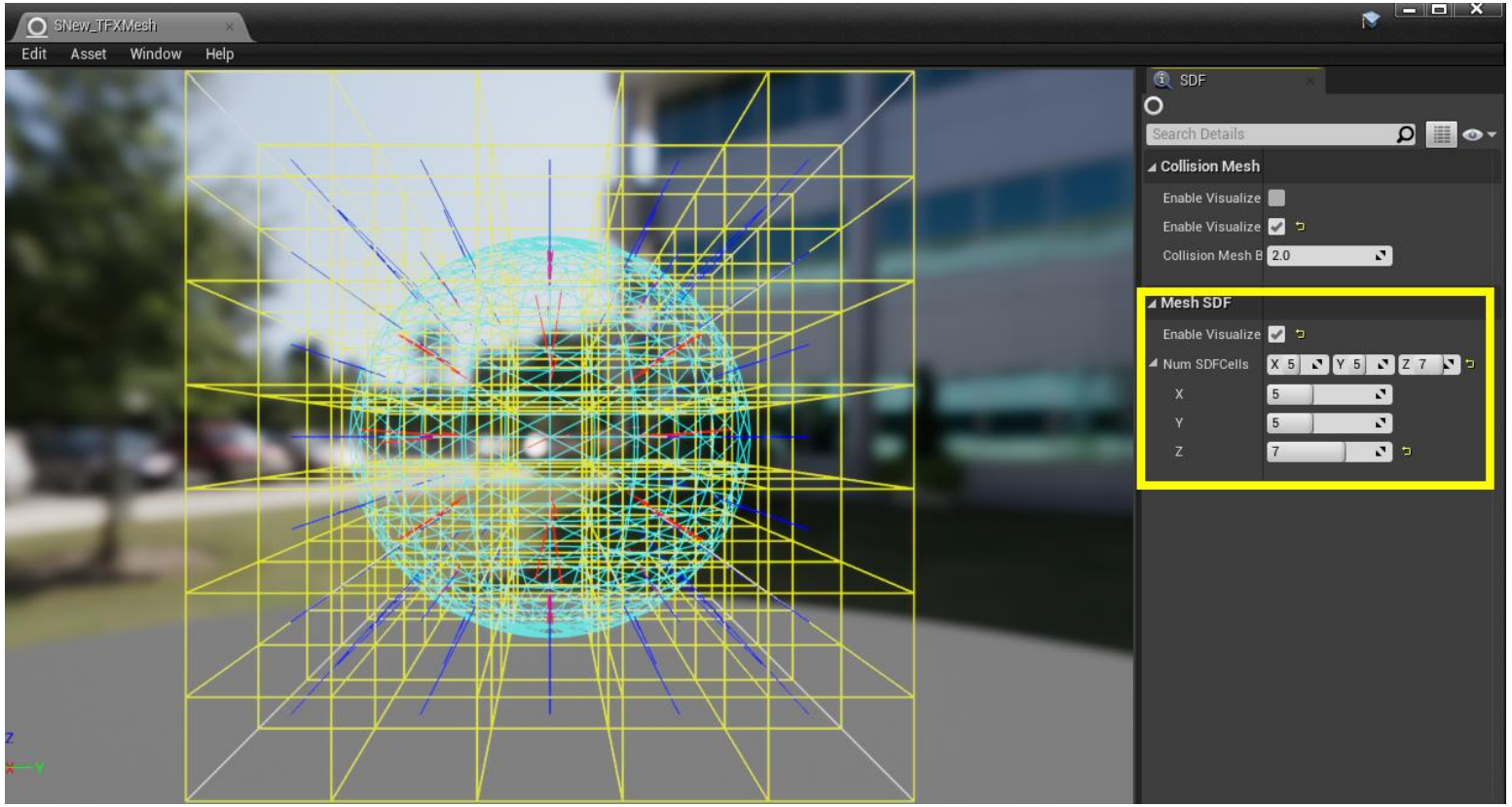
- To check whether it has been fixed,

also need to reproduce it

- Try to make a video by UE4 Sequencer

Improved <Animation data used for simulation lags a frame behind UE4 Skinned Mesh animation data>

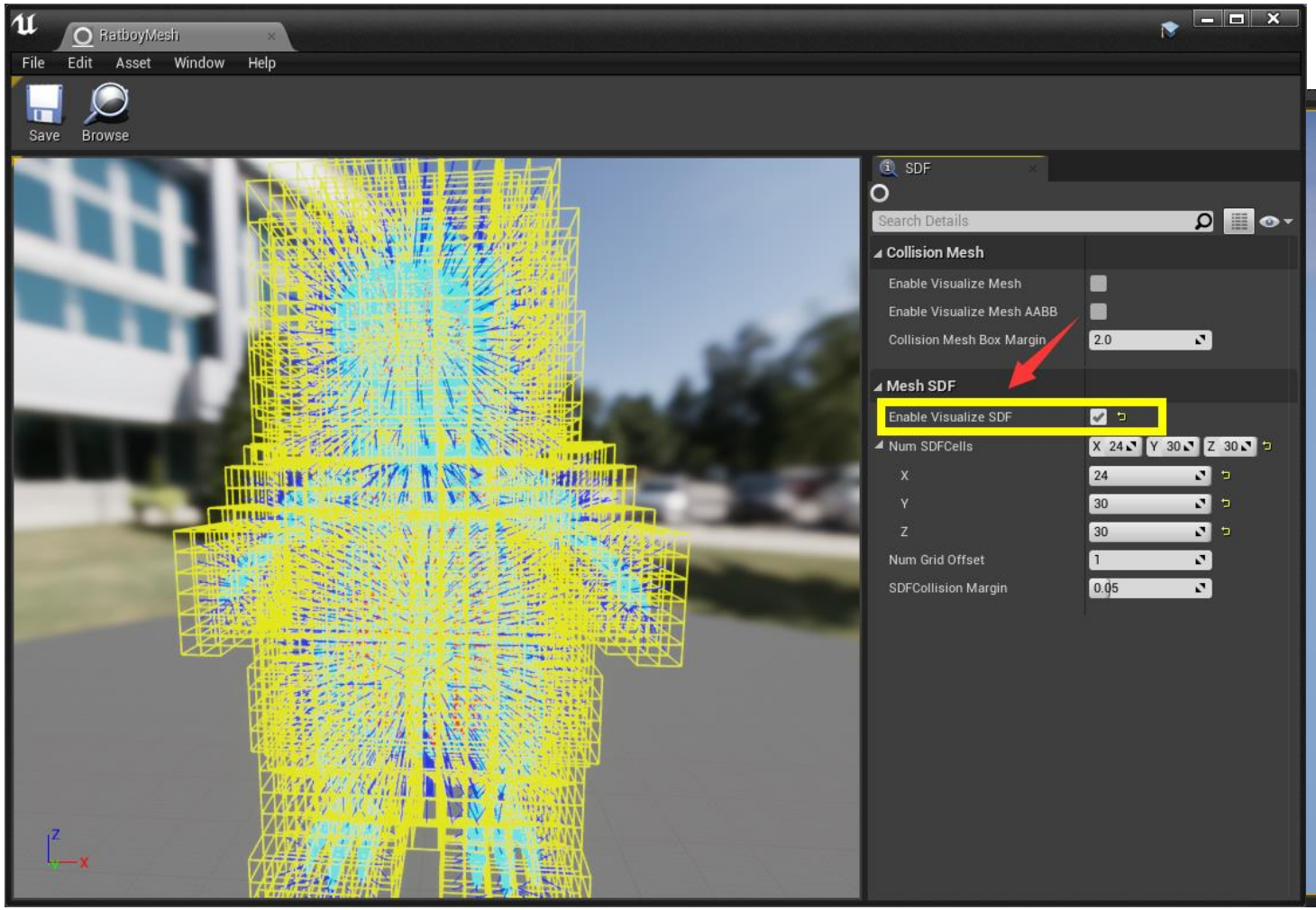
Validate the SDF Computation

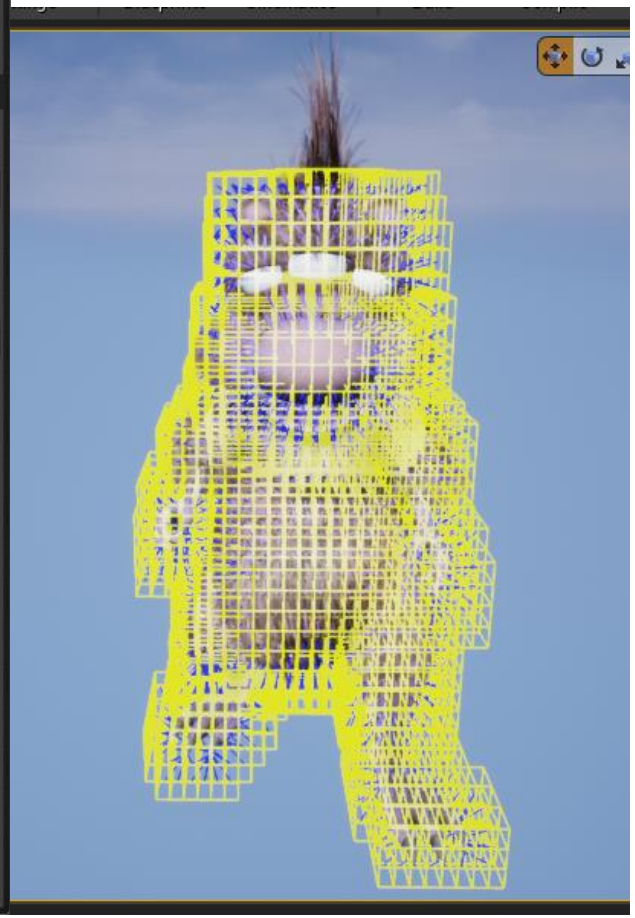


- Implement SDF Visualization
- SDF Visualization Algorithm:

<Per Voxel Linked List>,
it's a 3D version of <Per Pixel Linked List>

Validate the SDF Computation





SDF Avoid Penetration

SDF based Interaction with hands

New Resource Management to implement Editors

- TressFXComponent
  - InitResources
    - Gen TressFXGroupInstance and Register in TressFXManager
  - ReleaseResources
    - Unregister in TressFXManager and delete TressFXGroupInstance

- TressFXGroupInstance
  - WorldType – to handle different kinds of Editors
  - GuidesRestResources – static data
  - GuidesDeformedResources – do the simulation
  - StrandsResources – do the interpolation and rendering

Rendering Improvements

- ShortCut Simplification
- Directly use UE4 Hair Shading Model
- DeepShadow Optimization
- Strands RootUV and StrandUV Feature
- Support TAA

ShortCut Simplification

- Transparent
  - K-Layers Default Blend Formula (K=3)

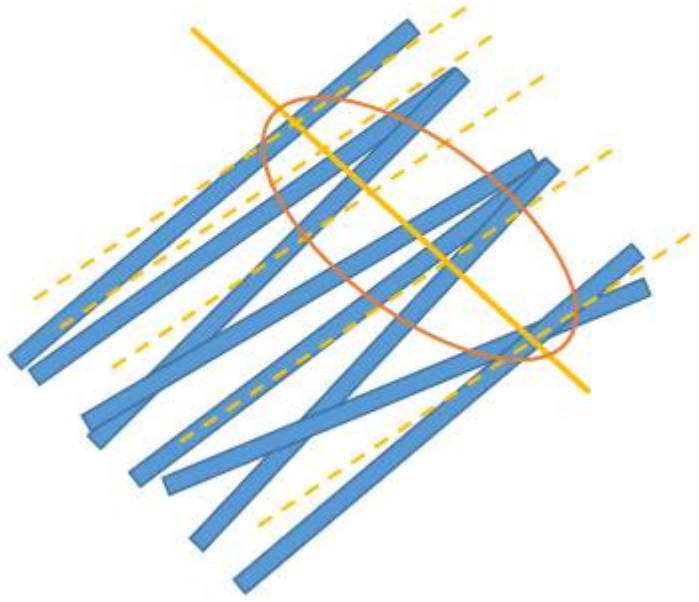
$$C_{final} = \alpha_0 C_0 + (1-\alpha_0)(\alpha_1 C_1 + (1-\alpha_1)(\alpha_2 C_2 + (1-\alpha_2)C_3))$$
$$C_{final} = \alpha_0 C_0 + (1-\alpha_0)\alpha_1 C_1 + (1-\alpha_0)(1-\alpha_1)\alpha_2 C_2 + (1-\alpha_0)(1-\alpha_1)(1-\alpha_2)C_3$$

  - K-Layers ShortCut Blend Formula (K=3)

$$C_{final} = \frac{(\alpha_0 C_0 + \alpha_1 C_1 + \alpha_2 C_2)}{(1+\alpha_0+\alpha_1+\alpha_2)}(1-(1-\alpha_0)(1-\alpha_1)(1-\alpha_2)\cdots(1-\alpha_n))$$
$$+(1-\alpha_0)(1-\alpha_1)(1-\alpha_2)\cdots(1-\alpha_n)C_{opaque}$$

  - ShortCut Simplification just shading the 1st Layer

DeepShadow Optimization



- New Shading Model needs Hair Count info to do Global Scattering
- Hundreds of layers texture IO may happen to many pixels
- PPLL to randomly sample 20 layers to approximate

Strands RootUV and StrandUV Feature

TressFXVertexFactoryCommon.ush | TressFXVertexFactory.ush | LightRendering.cpp | DeferredShadingRenderer.cpp | TressFXSDFComponent.cpp | TressFX

```
42
43  struct FVertexFactoryInterpolantsVSToPS
44  {
45      TANGENTTOWORLD_INTERPOLATOR_BLOCK
46
47  #if INTERPOLATE_VERTEX_COLOR
48      half4   Color : COLOR0;
49  #endif
50
51  #if USE_INSTANCING
52      // x = per-instance random, y = per-instance fade out amount, z = hi
53      float4  PerInstanceParams : COLOR1;
54  #endif
55
56  #if NUM_TEX_COORD_INTERPOLATORS
57      float4  TexCoords[(NUM_TEX_COORD_INTERPOLATORS+1)/2]    : TEXCOORD0;
58  #elif USE_PARTICLE_SUBUVS
59      float4  TexCoords[1] : TEXCOORD0;
60  #endif
61
```

mmon.ush | TressFXVertexFactory.ush | LightRendering.cpp | DeferredShadingRenderer.cpp | TressFXSDFComponent.cpp | Tress

```
Interpolants = (FVertexFactoryInterpolantsVSToPS)0;

const FVertexInfo VertexInfo = GetVertexInfo(Input);
uint StrandID = VertexInfo.VertexIndex / TressFXVF_NumVerticesPerSt
StrandID = TressFXVF_StrandsIDBuffer[StrandID];
float2 RootUV = TressFXVF_RootUVBuffer[StrandID].xy;
float2 StrandUV = float2(VertexInfo.IsLeft ? 0 : 1.0, float(VertexI

#if NUM_TEX_COORD_INTERPOLATORS > 0
    Interpolants.TexCoords[0].xy = RootUV;
    #if NUM_TEX_COORD_INTERPOLATORS > 1
        Interpolants.TexCoords[0].zw = StrandUV;
    #endif
#endif
```

Local Windows Debugger

ommon.ush | TressFXVertexFactory.ush | LightRendering.cpp | DeferredShadingRenderer.cpp | TressFXSDFComponent.cpp

```
ndif

Result.TwoSidedSign = 1;
Result.PrimitiveId = GetPrimitiveId(Interpolants);
Result.HairPrimitiveId = Interpolants.HairPrimitiveId;
Result.HairPrimitiveUV = float2(Interpolants.HairPrimitiveUV);

#if NUM_TEX_COORD_INTERPOLATORS > 0
    Result.TexCoords[0] = Interpolants.TexCoords[0].xy;
    #if NUM_TEX_COORD_INTERPOLATORS > 1
        Result.TexCoords[1] = float2(Interpolants.HairPrimitiv
    #endif
#endif
```

New Mainflow of TressFX Simulation and Rendering

- FDeferredShadingSceneRenderer::Render
  - InitViews
  - ❖ TressFXSimulation
    - ❖ ProcessSDFGen
    - ❖ ProcessGuideSimulation
    - ❖ ProcessStrandsInterpolation
  - RenderBasePass
  - ❖ RenderTressFXPrePass
    - ❖ DeepShadow
  - ❖ RenderTressFXBasePass
    - ❖ Translucency/MaterialData/VelocityBuffer
  - ❖ RenderLights
  - ❖ RenderEnvLight

UE4 TressFX 5.0 Improvements

- Maya Python Exporter
  - Both Strands and Collision Mesh vertices support the number of binding bones up to 16
  - Strands/Collision Mesh should be exported in the right initial position and keyframe
- Asset Editor & Visualization Tools
  - Support LevelMap/Asset/Blueprint Editors and fixed previous bugs
  - Visualization tools to check whether asset is correct, like tangents, collision mesh
- Simulation
  - Improved <Animation data used for simulation lags a frame behind UE4 Skinned Mesh animation data>
  - Improved Simulation Editor and implement SDF Editor
  - SDF BoundingBox is computed on GPU side
  - Improved SDF feature according to developer feedback
- Rendering
  - Support TAA/SkyLight/StrandsUV
  - Support Marschner ShadingModel
  - Support Cast/Receive Shadows
  - Improved ResourceManagement which is very important to implement Editors

UNREAL ENGINE INTEGRATION

Integration

- Asset
- Engine
- Snapshots
- Tools
- README.md

Asset › Content ›

名称

- Ratboy
- NewWorld.umap

Asset › Content › Ratboy

名称

- Ratboy.uasset
- Ratboy_Anim.uasset
- Ratboy_Eyes_MAT.uasset
- Ratboy_PhysicsAsset.uasset
- Ratboy_Skeleton.uasset
- RatboyBP.uasset

Engine ›

名称

- Plugins
- Shaders
- Source

Plugins › Runtime › TressFX › Source ›

名称

- TressFXCore
- TressFXEditor
- TressFXImportTranslator

Shaders › Private

名称

- TressFX
- DeferredLightPixelShadersTFX.usf
- DeferredLightVertexShadersTFX.usf
- MaterialTemplate.ush

LightRendering.cpp | DeferredShadingRenderer.cpp | TressFXSDFComponent.cpp | TressFXI

UE4 | (Global Scope)

```
24  // AMD TressFX BEGIN
25  #include "TressFX/TressFXRendering.h"
26  // AMD TressFX END
```

Source › Runtime › Renderer › Private ›

名称

- TressFX
- DeferredShadingRenderer.cpp
- DeferredShadingRenderer.h
- IndirectLightRendering.cpp
- LightGridInjection.cpp
- LightRendering.cpp
- SceneRendering.cpp
- SceneRendering.h
- SceneVisibility.cpp
- ShadowRendering.cpp
- ShadowRendering.h

Integration <UE4.26.2>

- Directly Copy and Replace the Engine folder to your local Engine folder
  - Engine/Plugins
  - Engine/Shaders
  - Engine/Source
- Double Click GenerateProjectFiles.bat
- Recompile UE4.sln
- Create a new Project with StarterContent
- Enable the Plugin TressFX 5.0 in PluginManager
- Copy the Asset/Content folder to the new project's Content folder
  - Content/Ratboy
  - Content/NewWorld
- Startup the project and Open NewWorld LevelMap
- The Lights(except SkyLight) in your LevelMap should enable CastDeepShadow before TressFX Render correctly

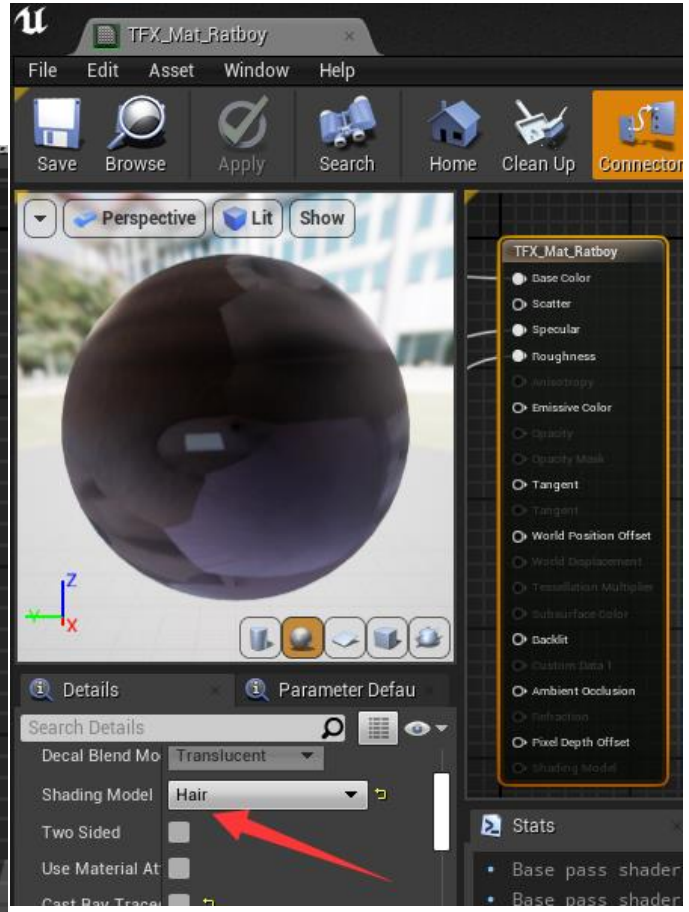
HOW TO USE IT

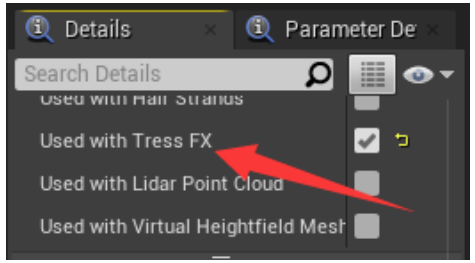
How to Use in UE4

- TressFX Material
- TressFX/TressFXMesh Asset Editor
- TressFX Blueprint Editor

TressFX Material

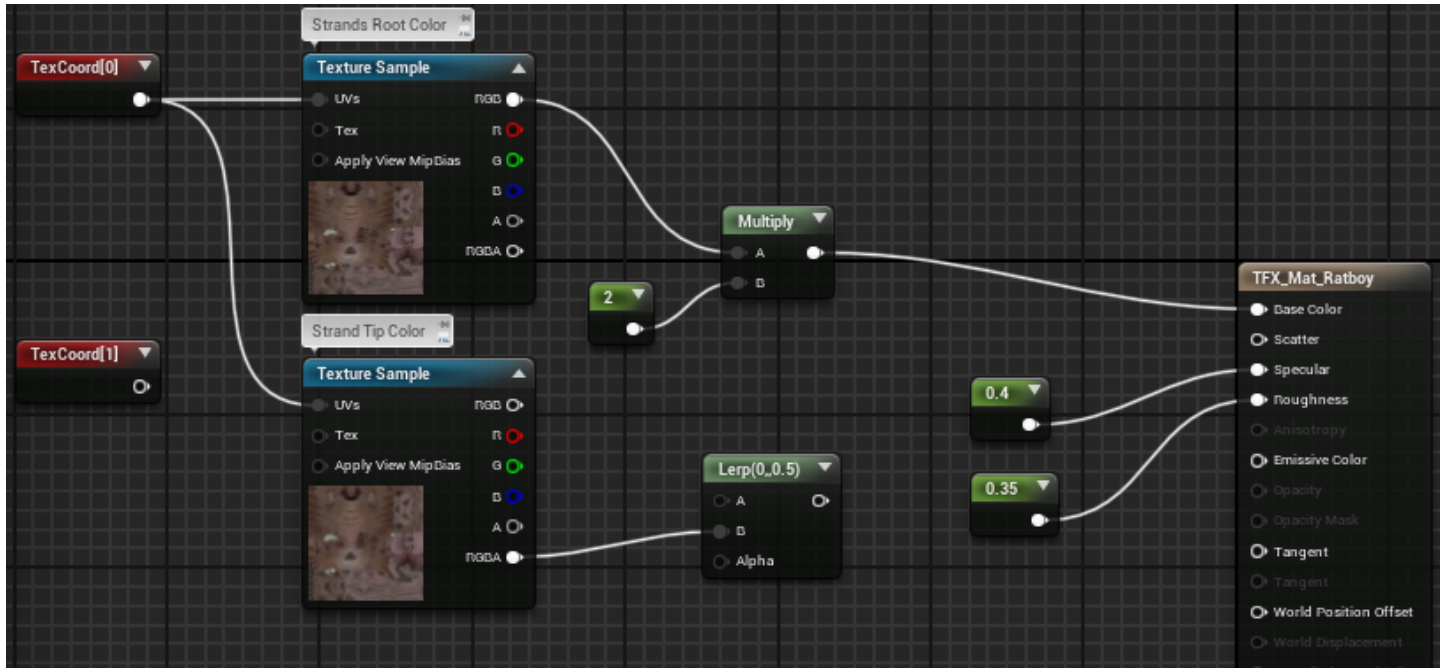
- Select <Result node of the Material>
- Set <Shading Model> to <Hair>
- Enable <Used with TressFX>



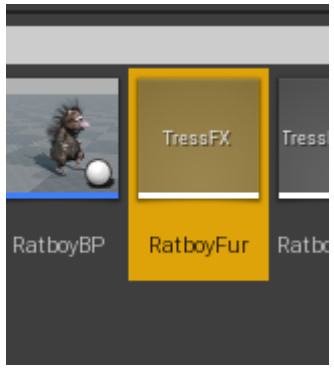


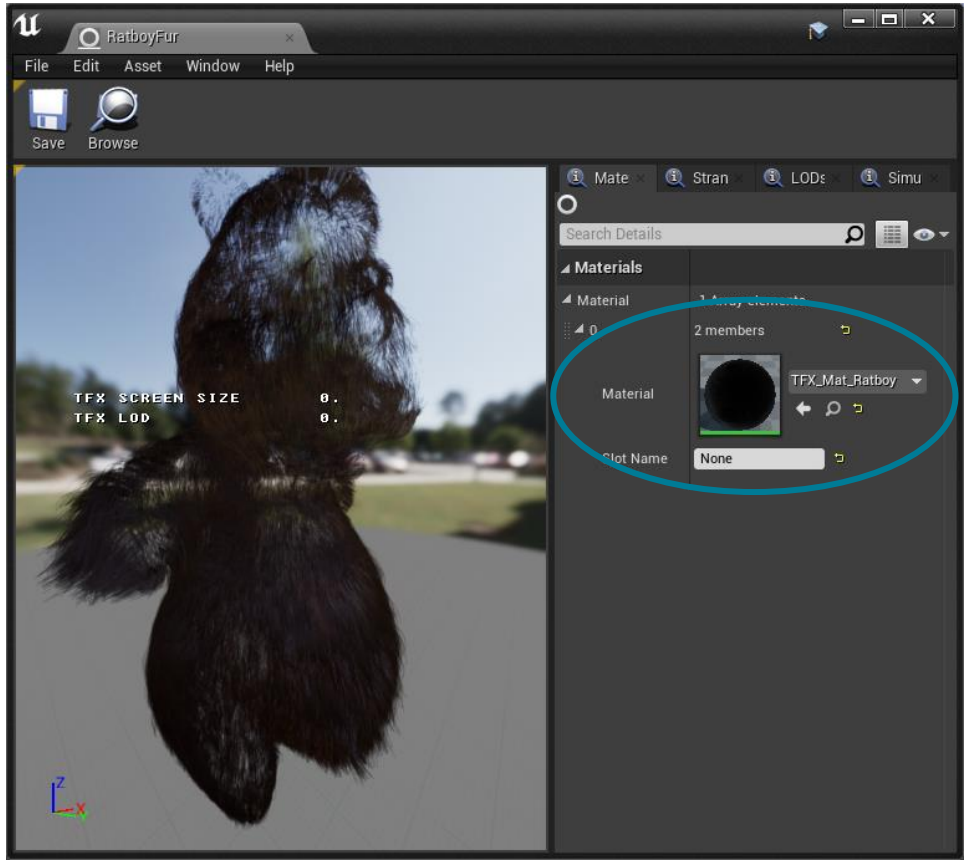
TressFX Material

- Recommend to Name your TressFX Material: TFX_Mat_*
- TexCoord[0] is RootUV
- TexCoord[1] is StrandUV



TressFX Asset Editor

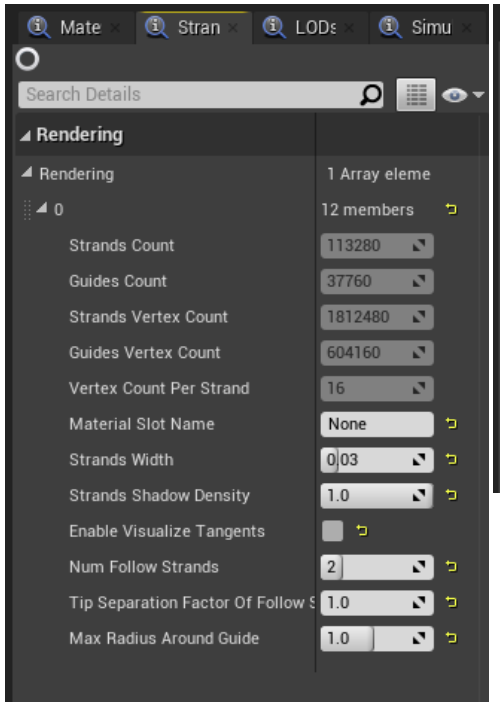


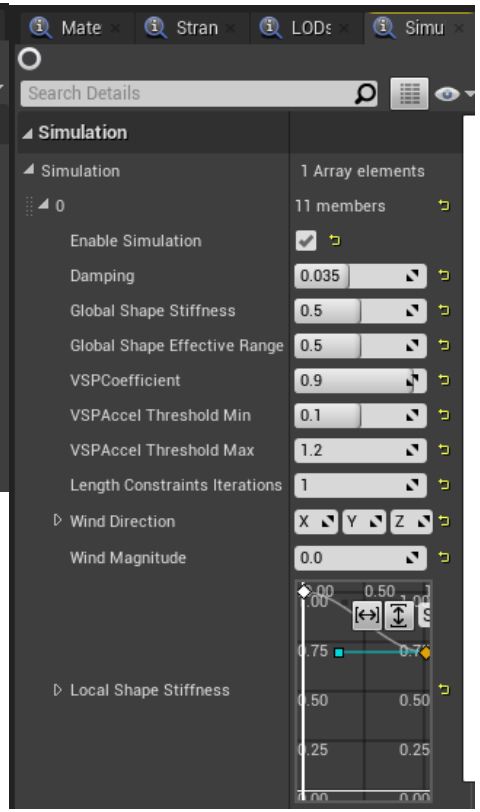


- Double Click RatboyFur Asset Open the AssetEditor
- Material Panel
- Choose and Set TFX_Mat_Ratboy Material

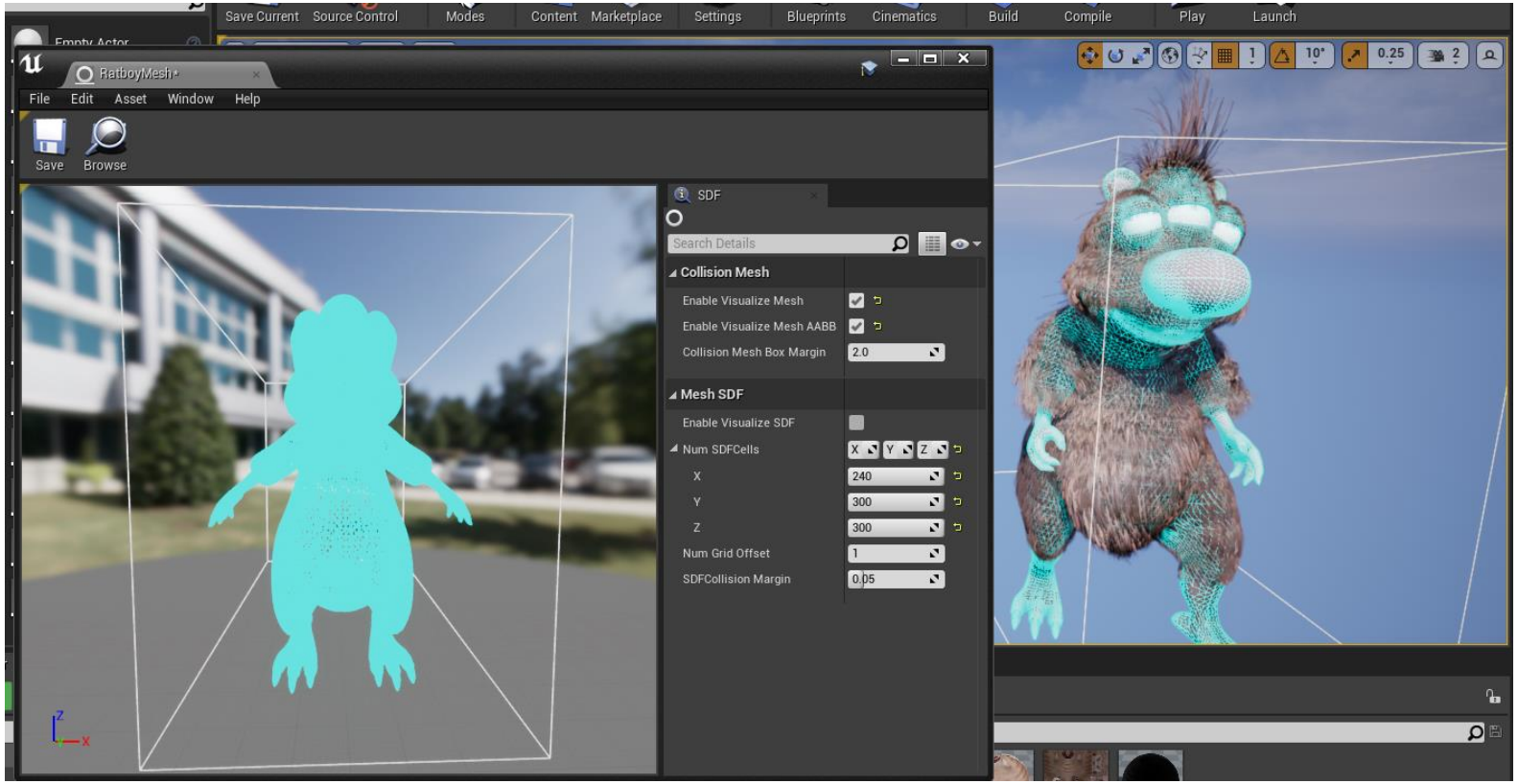
TressFX Asset Editor

- Strands Panel
- LOD Panel
- Simulation Panel





TressFXMesh Asset Editor

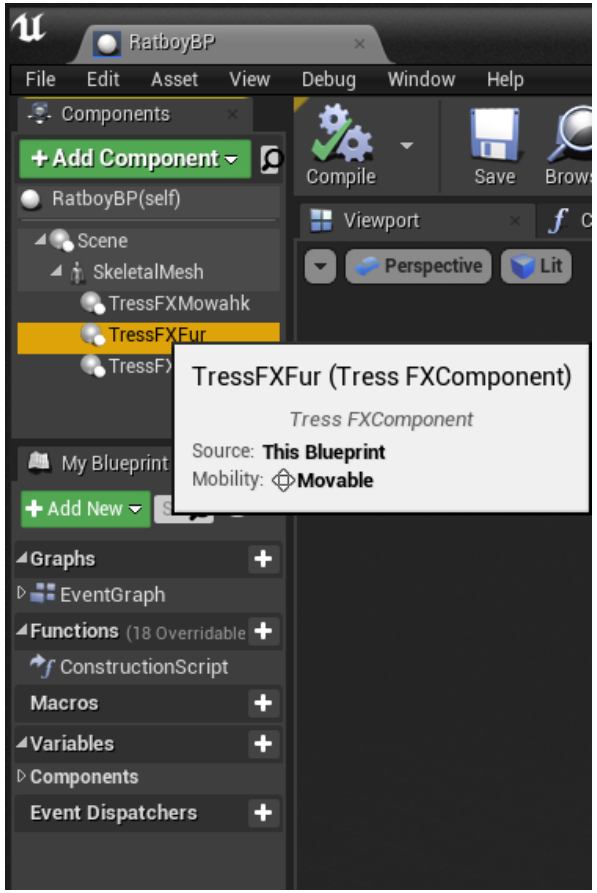


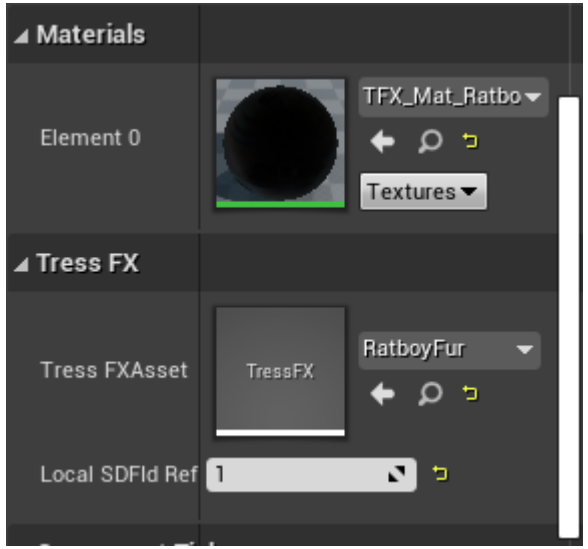
- Double Click RatboyMesh Asset
- Open the TressFXMesh AssetEditor
- Collision Mesh
  - Enable Visualize Mesh
  - Enable Visualize MeshAABB

TressFX Blueprint Editor

- Attach TressFXComponent to SkeletalMesh
- Set TressFX Asset
- Set Local SDFId Ref to Enable

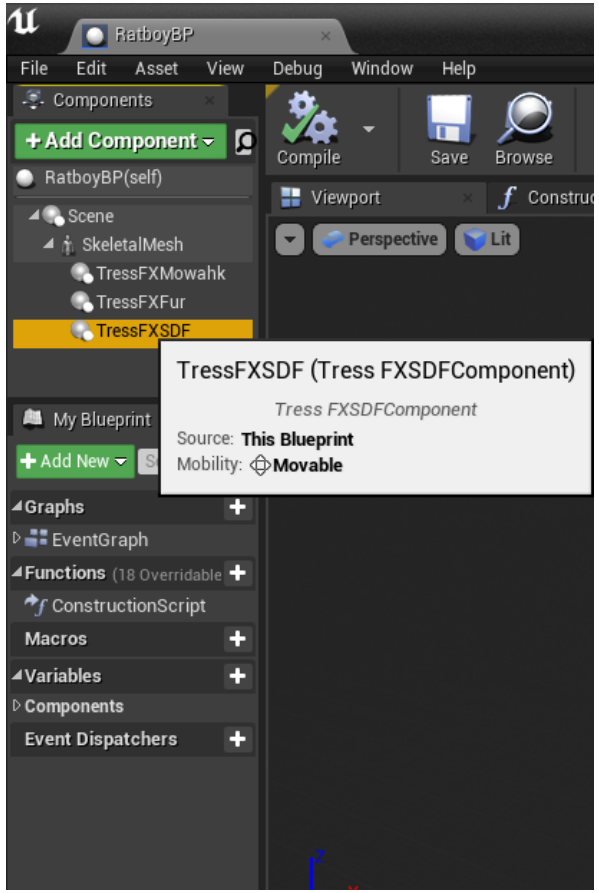
  SDF Collision Feature

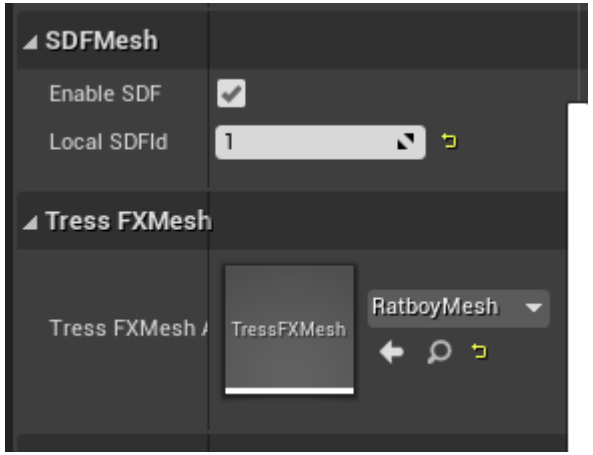




TressFX Blueprint Editor

- Attach TressFXSDFComponent to SkeletalMesh
- Set TressFXMesh Asset
  - Enable SDF
  - Set Local SDFId



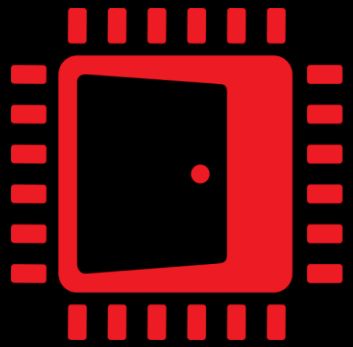


How to Use in UE4

FUTURE WORK

Future Work

- Support UE4.27 / UE5
- Support BlendShape Vertex Animation
- Research more about Rendering Optimization



AMD GPUOpen

RADEON

AMD

DISCLAIMER

- The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18.
-
- Other product names used in this publication are for identification purposes only and may be trademarks of their respective owners.
- Epic Games is the owner of the Unreal® Engine and owns the right of UE4 source codes and editors.