Fbx Artist Instructions

↓

AssetPipeline

(Creates the runtime assets while

performing validity checks to make

sure source assets fit the rules)

↓

ozz Runtime

↑ ↑

The Forge Gladiator

Mesh Optimization Artist Instructions

↓

AssetPipeline

(Creates the runtime assets while

performing validity checks to make

sure source assets fit the rules)

↓

Mesh Optimizer

↑ ↑

The Forge Gladiator

Artist Instructions

* Animated Objects
* Must be in FBX
* Create folder with the name of the animated object
* In this folder a single file entitled “riggedMesh.fbx”
* This file must have all the skeletal data required with the mesh it is bound to included as well
* No additional information/nodes that do not relate to the skin or skeleton
* No animation data, skeleton and mesh must be in bind pose. Before exporting, select the mesh and go to Rigging->Skin->Go to Bind Pose
* In this folder a subfolder entitled “animations”
* Each animation must be in its own separate fbx file
* No mesh/skin data or other extraneous data besides skeleton and animation data should be present
* Mesh Optimization
* Must be in glTF
* Create folder to store un-optimized models
* Place all models that need to be optimized in this folder
* Each .gltf file should have an accompanying .bin file for glTF 2.0
* Create folder to store optimized models
* This file will store the .gltf and .bin files of the optimized models

Programming documentation

**Library interface:**

struct ProcessAssetsSettings  
{  
 bool quiet; // Only output warnings.  
 bool force; // Force all assets to be processed.  
 uint minLastModifiedTime; // Force all assets older than this to be processed.

uint quantizationPositionBits; // N value for N-Bit position quantization.

uint quantizationNormalBits; // N value for N-Bit normal and tangent quantization.

uint quantizationTexBits; // N value for N-Bit texture coordinate quantization.  
  
};  
   
class AssetPipeline  
{  
public:  
 static bool ProcessAnimations(const char\* animationDirectory, const char\* outputDirectory,

ProcessAssetsSettings\* settings);

static bool CreateRuntimeSkeleton(const char\* skeletonAsset, const char\* skeletonName,

const char\* skeletonOutput, ozz::animation::Skeleton\* skeleton,

ProcessAssetsSettings\* settings);

static bool CreateRuntimeAnimation(const char\* animationAsset,

ozz::animation::Skeleton\* skeleton, const char\* skeletonName,

const char\* animationName, const char\* animationOutput,

ProcessAssetsSettings\* settings);

static bool ProcessMeshes(const char\* meshDirectory, const char\* outputDirectory,

ProcessAssetsSettings\* settings);

};

struct TFXAsset

{

// Hair data from \*.tfx

eastl::vector<float4> mPositions;

eastl::vector<float2> mStrandUV;

eastl::vector<float4> mRefVectors;

eastl::vector<float4> mGlobalRotations;

eastl::vector<float4> mLocalRotations;

eastl::vector<float4> mTangents;

eastl::vector<float4> mFollowRootOffsets;

eastl::vector<int> mStrandTypes;

eastl::vector<float> mThicknessCoeffs;

eastl::vector<float> mRestLengths;

eastl::vector<int> mTriangleIndices;

int mNumVerticesPerStrand;

int mNumGuideStrands;

};

struct TFXVertex

{

float3 mPosition;

float3 mNormal;

uint mBoneIndices[4];

float mBoneWeights[4];

};

struct TFXMesh

{

eastl::vector<eastl::string> mBones;

eastl::vector<TFXVertex> mVertices;

eastl::vector<uint> mIndices;

};

class AssetLoader  
{  
public:  
 static bool LoadSkeleton(const char\* skeletonFile, FSRoot root,

ozz::animation::Skeleton\* skeleton);

static bool LoadAnimation(const char\* animationFile, FSRoot root,

ozz::animation::Animation\* animation);

static bool LoadModel(const char\* modelFile, FSRoot root, Model\* model);

static bool ImportTFX(const char\* filename, FSRoot root, int numFollowHairs,

float tipSeperationFactor, float maxRadiusAroundGuideHair,

TFXAsset\* tfxAsset);

static bool ImportTFXMesh(const char\* filename, FSRoot root, TFXMesh\* tfxMesh);

static bool ImportTFXV3(File\* file, int numFollowHairs, TFXAsset\* tfxAsset);

};

**Errors and warnings:**

! AssetLoader::LoadModel only supports glTF 2.0 !

|  |  |
| --- | --- |
| Errors: | Warnings: |
| * Invalid usage of the application. | * No assets found. |
| * Failure to read, write or create files. | * Assets don’t follow artist instructions. |
| * Invalid skeleton. * Invalid .gltf file. | * Invalid animation. |

! Invalid skeletons throw errors because they’re required to draw skeletal mesh !

**Command line tool:**

/> AssetPipelineCmd.exe processanimations “<path to fbx assets>” “<output path>” [--quiet] [--force]

Animation Example:

/> AssetPipelineCmd.exe processanimations “<Animation/fbx>” “<Animation/output>”



/> AssetPipelineCmd.exe processmeshes “<path to un-optimized glTF>” “<output path>” "-posbits" "16" "- texbits" "16" "-normbits" "8"

Mesh Optimization Example:

/> AssetPipelineCmd.exe processmeshes “<Meshes/>”“<Meshes/optimized>”

**Code:**

The code should be self explanatory. The comments explain the basic flow of the program.

**References:**

Assimp: <http://assimp.sourceforge.net/lib_html/index.html>

<http://assimp.sourceforge.net/lib_html/data.html>

OZZ: <http://guillaumeblanc.github.io/ozz-animation/>

MeshOptimizer: <https://github.com/zeux/meshoptimizer>

**Known problems:**

* Skeletons and animations scaled from centimeters to meters (\* 0.01). This is only required for FBX files. If we want to support other formats this has to be changed.
* See: AssetPipeline.cpp - line: 411 & 568 & 571
* Boris (Unreal Paragon asset) rotated 90 degrees on x axis after putting it through the animation pipeline. Blender also doesn’t import model correctly. Rotating the model in either Maya or Blender causes severe problems in the animation. Makes it look like his nose, fingers and ankle are all broken.