Content Loader Introduction

by G.Paskaleva

Institute of Computer Graphics and Algorithms

Vienna University of Technology

Model Formats

- Quake II / III (md2 / md3, md4)
- Doom 3 (md5)
- FBX
- Ogre XML
- Collada (dae)
- Wavefront (obj)

…

Must-Have Geometry Information

- vertex coordinates
- vertex normals
- vertex texture coordinates
-
-
-
  -
  -
  -
  -
- face indices

Nice-To-Have Geometry Information

- vertex coordinates
- vertex normals
- vertex texture coordinates
- vertex tangent
- vertex bitangent
-
  -
  -
  -
  -
- face indices

Cool-To-Have Geometry Information

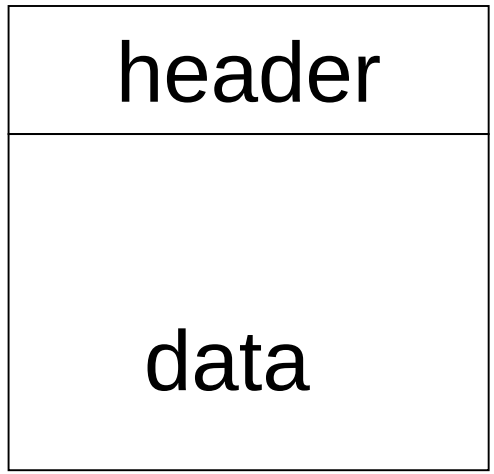
- vertex coordinates
- vertex normals
- vertex texture coordinates
- vertex tangent
- vertex bitangent
- custom vertex attributes, representing:
  - elasticity
  - bounciness
  - transparency
  - shininess…
- face indices

Also Transferable Are:

- transformation matrices
- animations
  - keyframe animations
  - skin-and-bone animation
  - morpher animations
- materials
- cameras
- lights
- global attributes

Quake II / Quake III

- **md2** was introduced by **id Software** when releasing Quake II in 1997
- **md3** was introduced by **id Software** when releasing Quake III Arena in 1999
- **md2** and **md3** are in a binary format
- not human readable



md2 Header

```
struct md2_header_t
{
  int ident;      /*magic number: "IDP2"*/
  int version;    /*version: must be 8*/
  int skinwidth;  /*texture width*/
  int skinheight; /*texture height*/
  int framesize;  /*size in bytes of a frame*/

  int num_skins;  /*number of skins*/
  int num_xyz;    /*num. vertices per frame*/
  int num_st;     /*number of texture coordinates*/
  int num_tris;   /*number of triangles*/
  int num_glcmds; /*number of opengl commands*/
  int num_frames; /*number of frames*/
```

…

md2 Header cont'd

```
  int ofs_skins;  /*offset skin data*/
  int ofs_st;     /*offset texture coord. data*/
  int ofs_tris;   /*offset triangle data*/
  int ofs_frames; /*offset frame data*/
  int ofs_glcmds; /*offset OpenGL command data*/
  int ofs_end;    /*offset end of file*/
};
```

source: quake2-3.21\qcommon\qfiles.h

http://www.idsoftware.com/business/techdownloads/

md2 Limits

- Quake II has some predefined limits so that dynamic memory does not need to be used:
  - triangles: 4096
  - vertices: 2048
  - texture coordinates: 2048
  - frames: 512
  - skins (textures) :32

md2 Peculiarities

- vertex coordinates are scaled to fit in 1 byte
  - they need to be rescaled before drawing!
- there are only 162 vertex normals to chose from
- texture coordinates depend on the size of the texture map
  - they need normalizing!

md3 Header

```
typedef struct {
    int  ident;
    int  version;
    char name[MAX_QPATH];      // model name
    int  flags;

    int  numFrames;
    int  numTags;
    int  numSurfaces;
    int  numSkins;

    int  ofsFrames;            // offset for first frame
    int  ofsTags;              // numFrames * numTags
    int  ofsSurfaces;          // first surface, others follow
    int  ofsEnd;               // end of file
} md3Header_t;
```


md3 Data Types

- in addition to the md2 data types there are also representations for:
  - surfaces (containing frames)
  - normals (in the range from 0 to 255)
    - encoded in a spherical coordinate system
    - 8 bit for latitude
    - 8 bit for longitude
  - shader (material)
  - tags (for aligning separate md3-objects)

md3 Limits

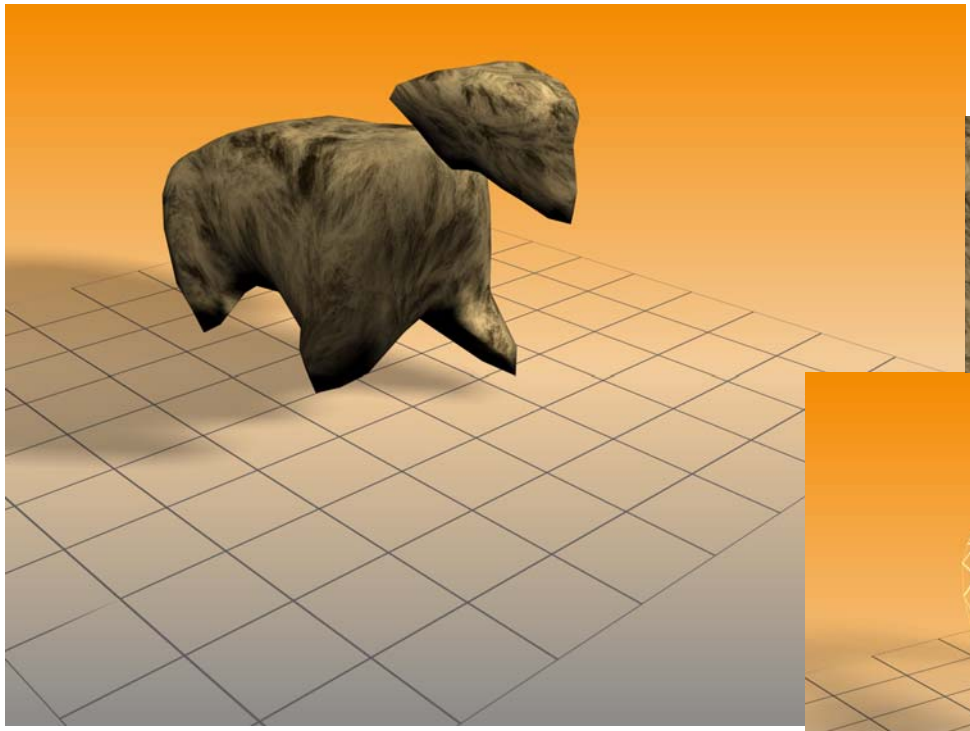
- Quake III has expanded its limits to:
  - triangles per surface: 8192
  - vertices per surface: 4096
  - shaders (materials) per surface: 256
  - frames per model: 1024
  - surfaces per model: 32
  - tags:16
  - LOD: 4 (applies to **md4**)

source: quake3-1.32\common\qfiles.h
http://www.idsoftware.com/business/techdownloads/

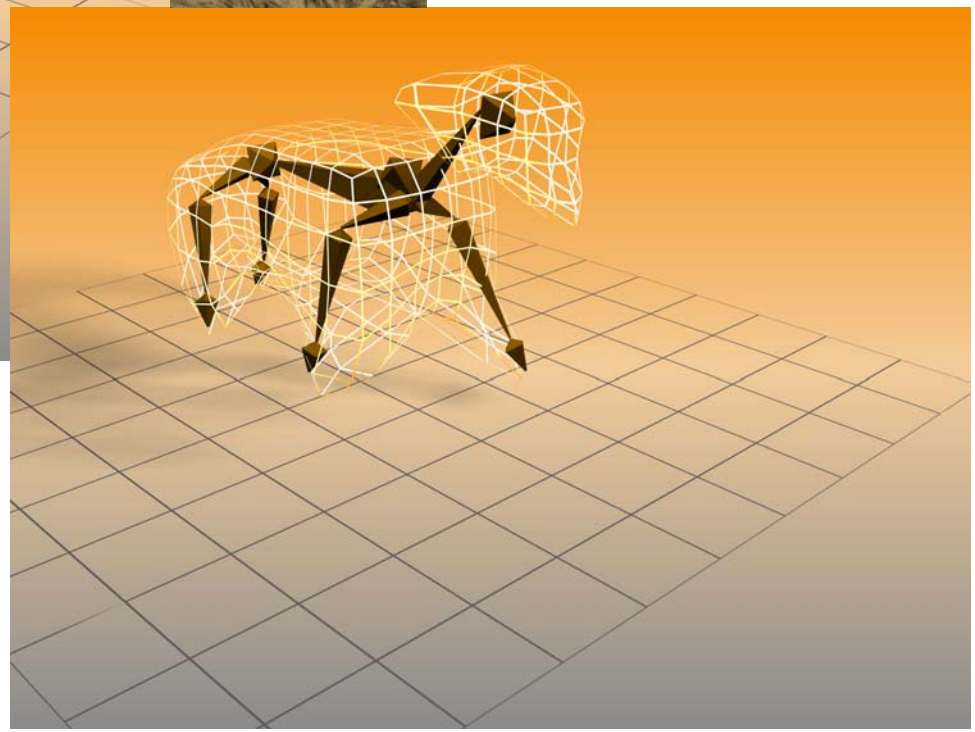
md4 for bone animations

- in addition to md3 contains representations for:
  - LOD
  - bones
  - weight of bones
  - vertices contain arrays of weight-references
  - surfaces contain bone- and weight-references
  - frames contain arrays of bone-references

Modeling Example



Skin-and-Bone Animation



17 bones

2 meshes (of 314 and 98 vertices)

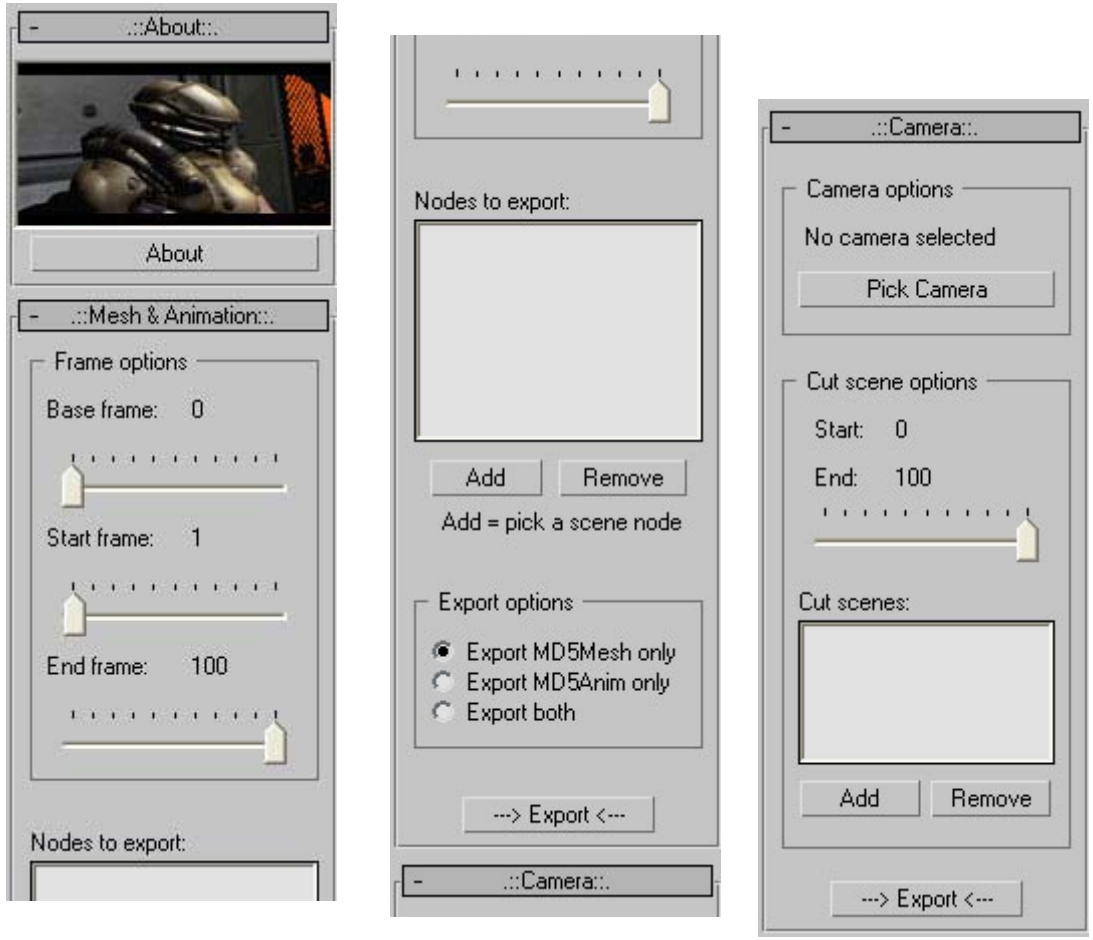
3 keyframes

Doom 3

- **md5** was introduced by **id Software** when releasing Doom 3 first person shooter in 2004
- **md5 files** are in ASCII format
- human readable

md5 Exporter “berserker”

- the exported meshes must have **“skin”** on top of the modifier stack



md5 Exported Files

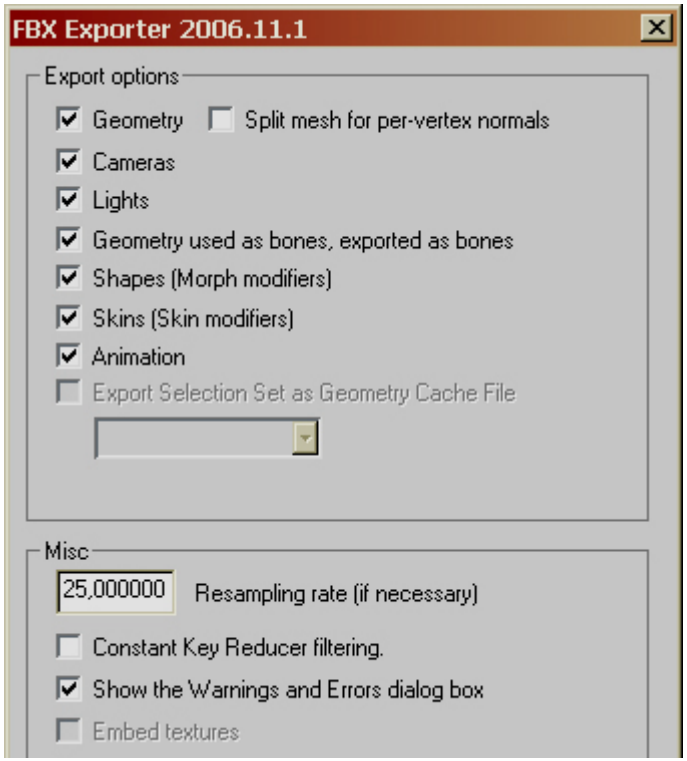
- 3 types of files result from the export:
  - *.**md5anim** for animations
  - *.**md5mesh** for geometry
  - *.**md5camera** for animated cameras

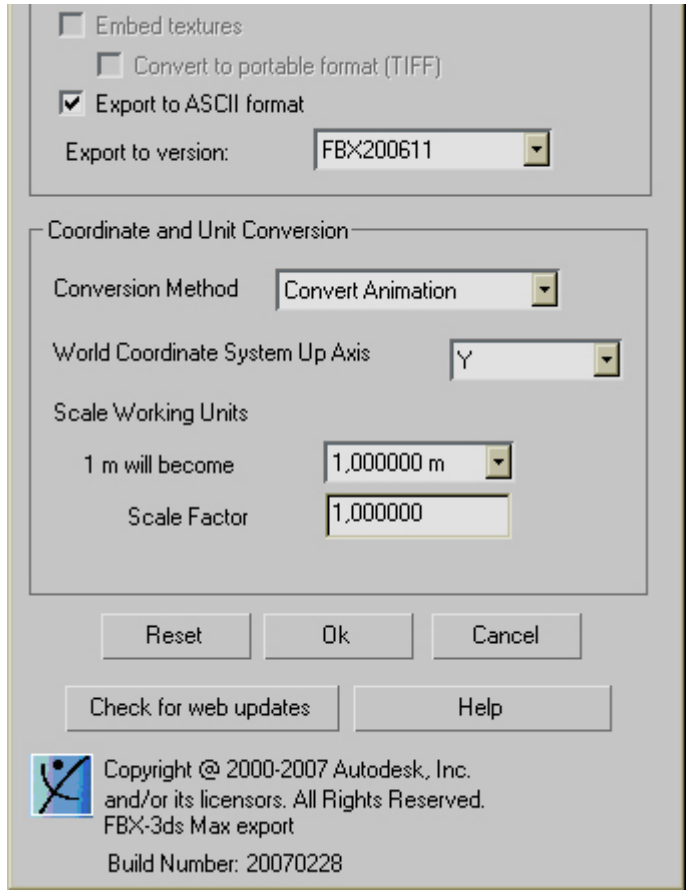
md5 Peculiarities

- vertex skinning
- uses quaternions for orientation
- does not export vertex normals!
- keyframe choice and interpolation “on foot” since not just the keyframes but every single frame is exported
- transformations in a hierarchy-chain are relative – if an object has 5 predecessors you need to perform 5 steps before determining its position in world space

FBX exporter

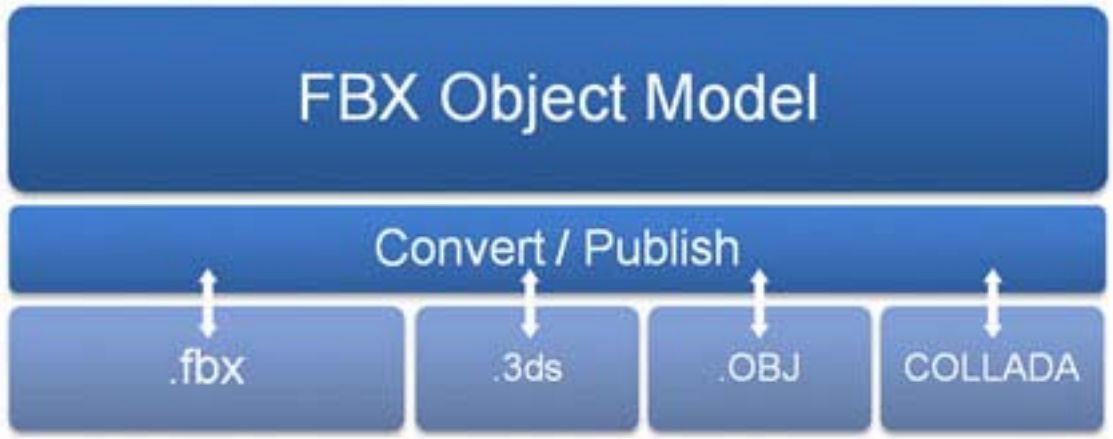
- part of the Autodesk FBX technology
- in binary format (with embedded textures)
- in ASCII format





FBX Object Model

- supported by Autodesk FBX SDK
- custom attributes definition in the SDK
- a FBX Application can contain more than one scene, but a FBX file can save only one scene
- an FBX scene is a graph



FBX file structure (1)

- general info section
- object definitions (comparable to a header)

```
Definitions:  {
    Version: 100
    Count: 65
    ObjectType: "Model" {
        Count: 35
    }
    ObjectType: "Geometry" {
        Count: 6
    }
    ObjectType: "Material" {
        Count: 2
    }
    ObjectType: "Texture" {
        Count: 1
    }
    ObjectType: "Video" {
        Count: 1
    }
    ObjectType: "Deformer" {
        Count: 19
    }
    ObjectType: "Pose" {
        Count: 1
    }
    ObjectType: "GlobalSettings" {
        Count: 1
    }
}
```


FBX file structure cont'd (2)

- object properties (> 60 !) including
  - transformations
  - axes
  - soft / rigid body properties …
- object relations (assignment to a category)
- object connections (parent-child hierarchy)
  - I.e. IK-chains
  - transformation constraints

FBX file structure cont'd (3)

- takes and animation section
  - models animation with the actual keyframes – I.e. if there were 3 keyframes over an animation of 400, there are 3 keys and not 400, the interpolation however is not specified
  - generic nodes animation
  - textures animation (?)
  - materials animation (?)
  - constraints animation
  - multiple takes possible but only one current
- settings (ambient fog, render settings etc.)

FBX sdk supported elements (1)

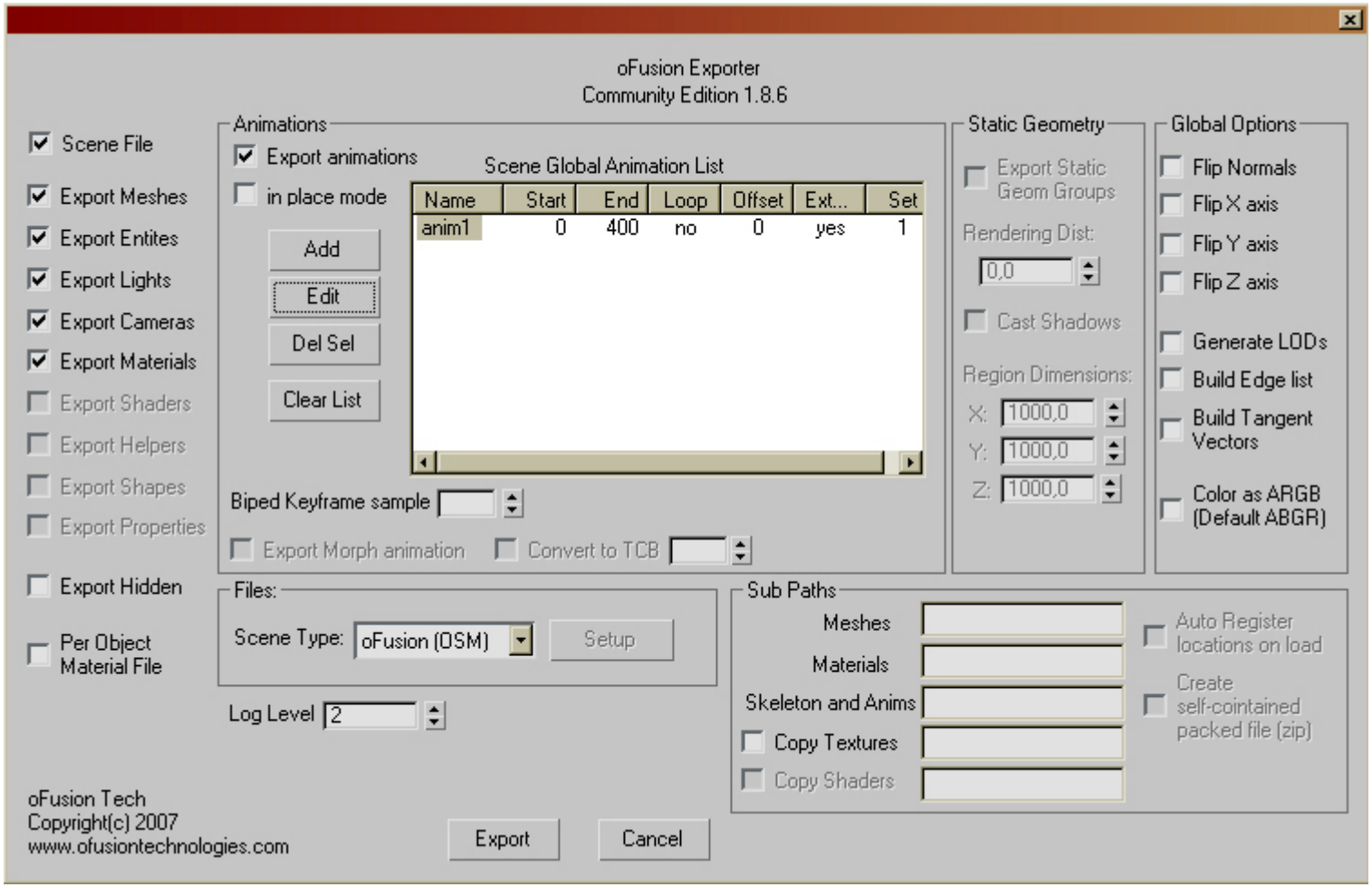
- geometry
  - mesh, patch, NURBS-surfaces and curves
  - texture mapping
  - material attributes mapping
  - vertex normals, colors, user-defined attributes
  - edge visibility
  - smoothing groups (interpolation of normals across neighboring polygons)
  - constraints over control points of geometry

FBX sdk supported elements cont'd (2)

- animation
  - by skinning, shapes
  - directly storing vertex animations of the control points of geometry in a **cache file**
- transformations
  - including constraints
- skeleton segments
  - root, limb, limb node
  - bind- and rest-pose for a list of nodes
- multiple (animated) cameras and lights

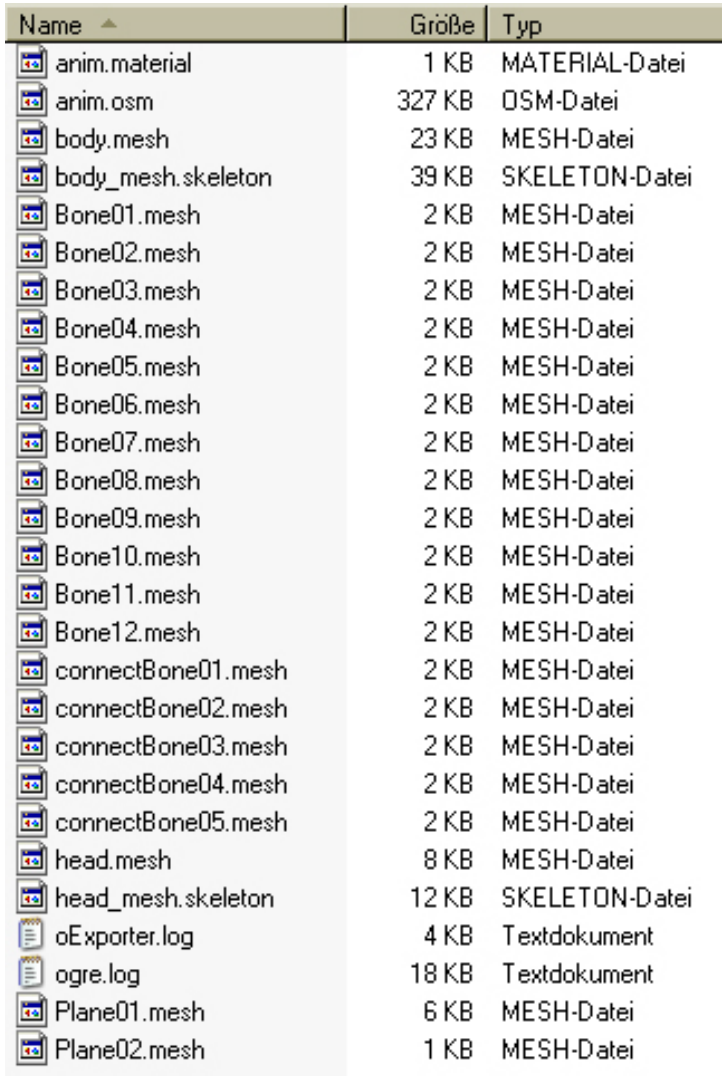
Ogre XML exporter

- for 3ds Max:



Ogre XML exported files (1)

- each object exported as a separate ***.mesh** file
- animations exported per skin-object as separate ***.mesh.skeleton** file
- all materials exported to a separate ***.material** file
- scene exported as a ***.osm** file



Name	Größe	Typ
anim.material	1 KB	MATERIAL-Datei
anim.osm	327 KB	OSM-Datei
body.mesh	23 KB	MESH-Datei
body_mesh.skeleton	39 KB	SKELETON-Datei
Bone01.mesh	2 KB	MESH-Datei
Bone02.mesh	2 KB	MESH-Datei
Bone03.mesh	2 KB	MESH-Datei
Bone04.mesh	2 KB	MESH-Datei
Bone05.mesh	2 KB	MESH-Datei
Bone06.mesh	2 KB	MESH-Datei
Bone07.mesh	2 KB	MESH-Datei
Bone08.mesh	2 KB	MESH-Datei
Bone09.mesh	2 KB	MESH-Datei
Bone10.mesh	2 KB	MESH-Datei
Bone11.mesh	2 KB	MESH-Datei
Bone12.mesh	2 KB	MESH-Datei
connectBone01.mesh	2 KB	MESH-Datei
connectBone02.mesh	2 KB	MESH-Datei
connectBone03.mesh	2 KB	MESH-Datei
connectBone04.mesh	2 KB	MESH-Datei
connectBone05.mesh	2 KB	MESH-Datei
head.mesh	8 KB	MESH-Datei
head_mesh.skeleton	12 KB	SKELETON-Datei
oExporter.log	4 KB	Textdokument
ogre.log	18 KB	Textdokument
Plane01.mesh	6 KB	MESH-Datei
Plane02.mesh	1 KB	MESH-Datei

Ogre XML exported files cont'd (2)

- ***.mesh** and ***.mesh.skeleton** are binary
- ***.mesh.xml** and ***.mesh.skeleton.xml** are human readable xml-files
- ***.material** is ASCII and contains for each mat.
  - ambient, diffuse and specular material color
- ***.osm** is ASCII and includes
  - the list of meshes with **attributes** including
    - parent entity
    - transformations
    - animations (keyframe index, transformation)

OgreXML *.mesh file scheme (1)

- shared geometry as a sequence of
  - vertex buffer as a sequence of
    - vertex
- sub-meshes as a sequence of
  - sub-mesh as a sequence of
    - textures
    - faces as a sequence of
      - face (1st defined by 3 vert., each after – by 1)
    - geometry as a sequence of
      - vertex buffer
    - bone assignments

OgreXML *.mesh file scheme cont'd (2)

- skeleton link
- bone assignments as a sequence of
  - vertex bone assignments
- LOD
- LOD generated as a sequence of
  - LOD-face list as a sequence of
    - face (1st defined by 3 vert., each after – by 1)
- poses as a sequence of
  - pose
- animations as a sequence of
  - tracks

OgreXML *.mesh file scheme cont'd (3)

- **vertex**
  - **position** (x,y,z) is mandatory
  - normal (x,y,z)
  - tangent (x,y,z,w)
  - binormal (actually bitangent) (x,y,z)
  - color diffuse ("r g b a" as a string)
  - color specular ("r g b a" as a string)
  - texture coordinates (s,t,r) for 3-dimensional textures

OgreXML *.mesh file scheme cont'd (4)

- **pose**
  - with **attributes** including
    - target mesh or sub-mesh
  - as a **sequence** of
    - pose offset with **attributes** including
      - index (which vertex)
      - x,y,z (offset amount)

OgreXML *.mesh file scheme cont'd (5)

- **tracks** as a sequence of
  - track
    - as a **sequence** of
      - keyframes as a sequence of
        - keyframe
    - with **attributes** including
      - target (mesh or sub-mesh)
      - type (morph or pose)

Keyframes are applicable for all tracks, but for morph tracks they contain positions, and for pose tracks they contain pose references.

OgreXML *.mesh file scheme cont'd (6)

- **mesh keyframe**
  - with **attribute**
    - time
  - as a **sequence** of
    - position
    - pose reference with **attributes**
      - pose index
      - influence

OgreXML *.mesh.skeleton file (1)

- bones as a sequence of
  - bone as a sequence of
    - position
    - rotation
    - scale
- bone hierarchy as a sequence of
  - bone parent with attributes
    - bone id (references bone)
    - parent id (references bone)

OgreXML *.mesh.skeleton cont'd (2)

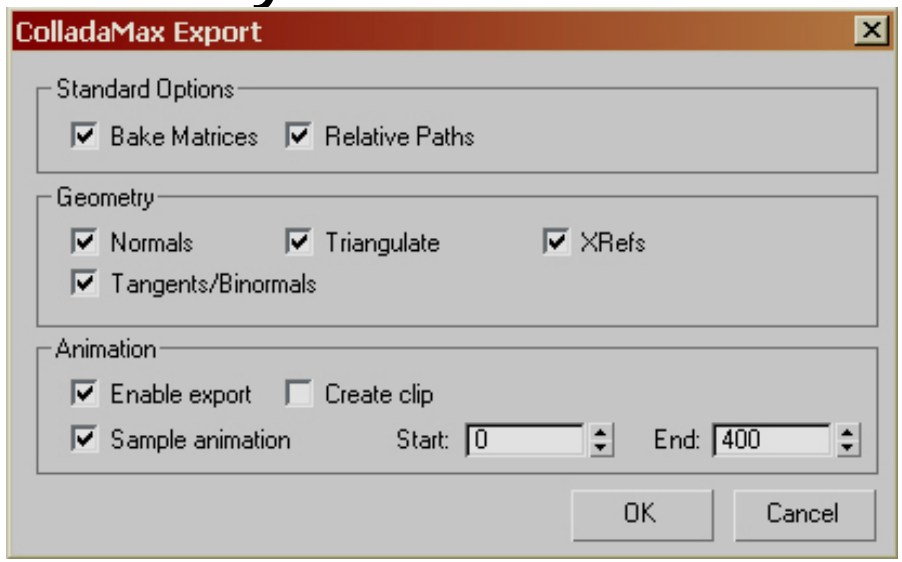
- animations as a sequence of
  - animation as a sequence of
    - tracks
      - with **attribute**
        - bone
      - as a **sequence** of
        - keyframes
- animation links as a sequence of
  - animation link with attributes
    - skeleton name (references a skeleton file)
    - scale

OgreXML *.mesh.skeleton cont'd (3)

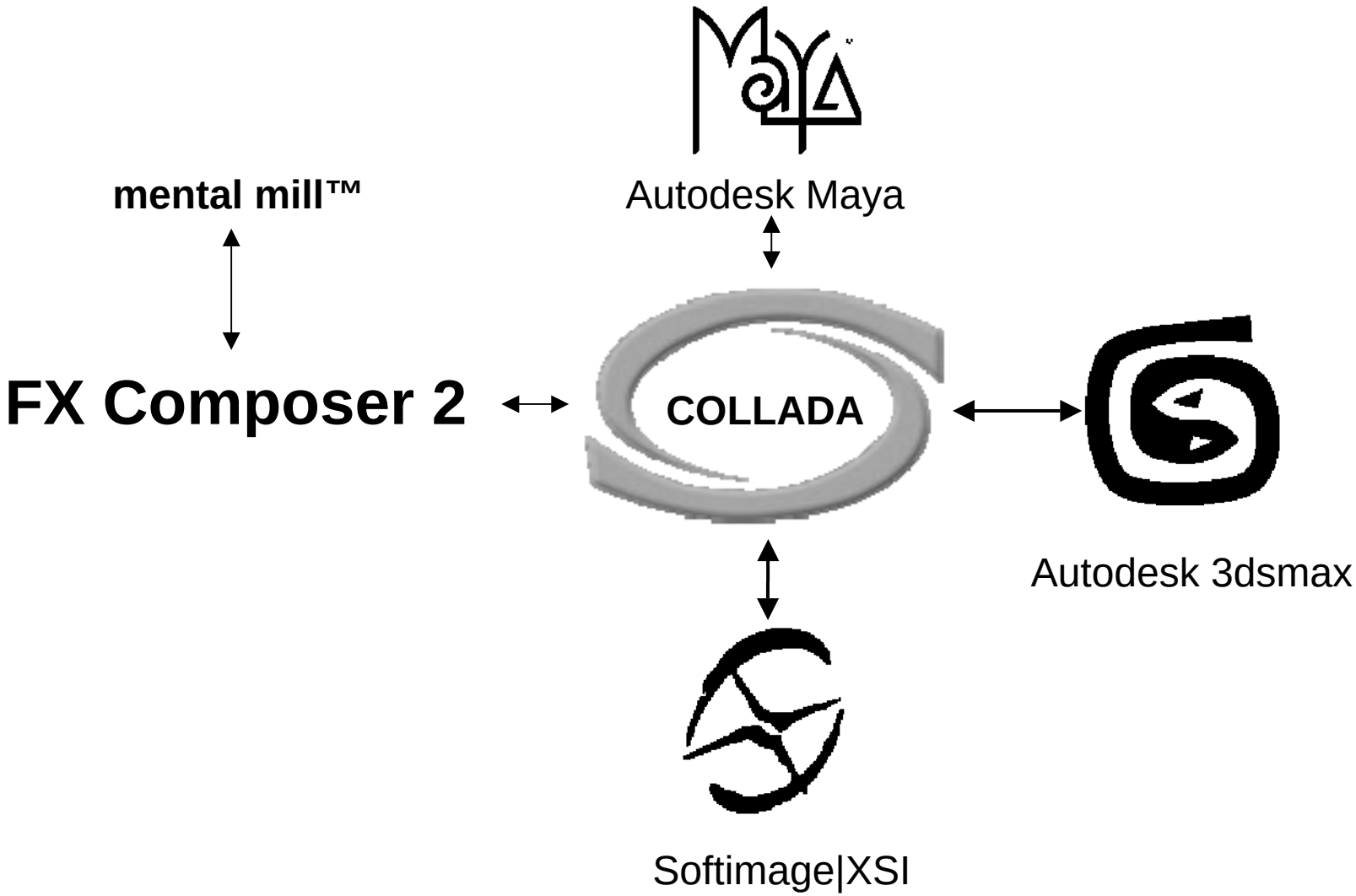
- **skeleton keyframe**
  - with **attribute**
    - time
  - as a **sequence** of
    - translate
    - rotate
    - scale

Collada Digital Asset Exchange

- **COLLA**borative **D**esign **A**ctivity for establishing an interchange file format for interactive 3D applications.
- latest version 1.5.0 from 2008
- adopted as **industry standard** by The Khronos Group since January 2006
- the exported XML file:
  - ***.DAE**



Collada support



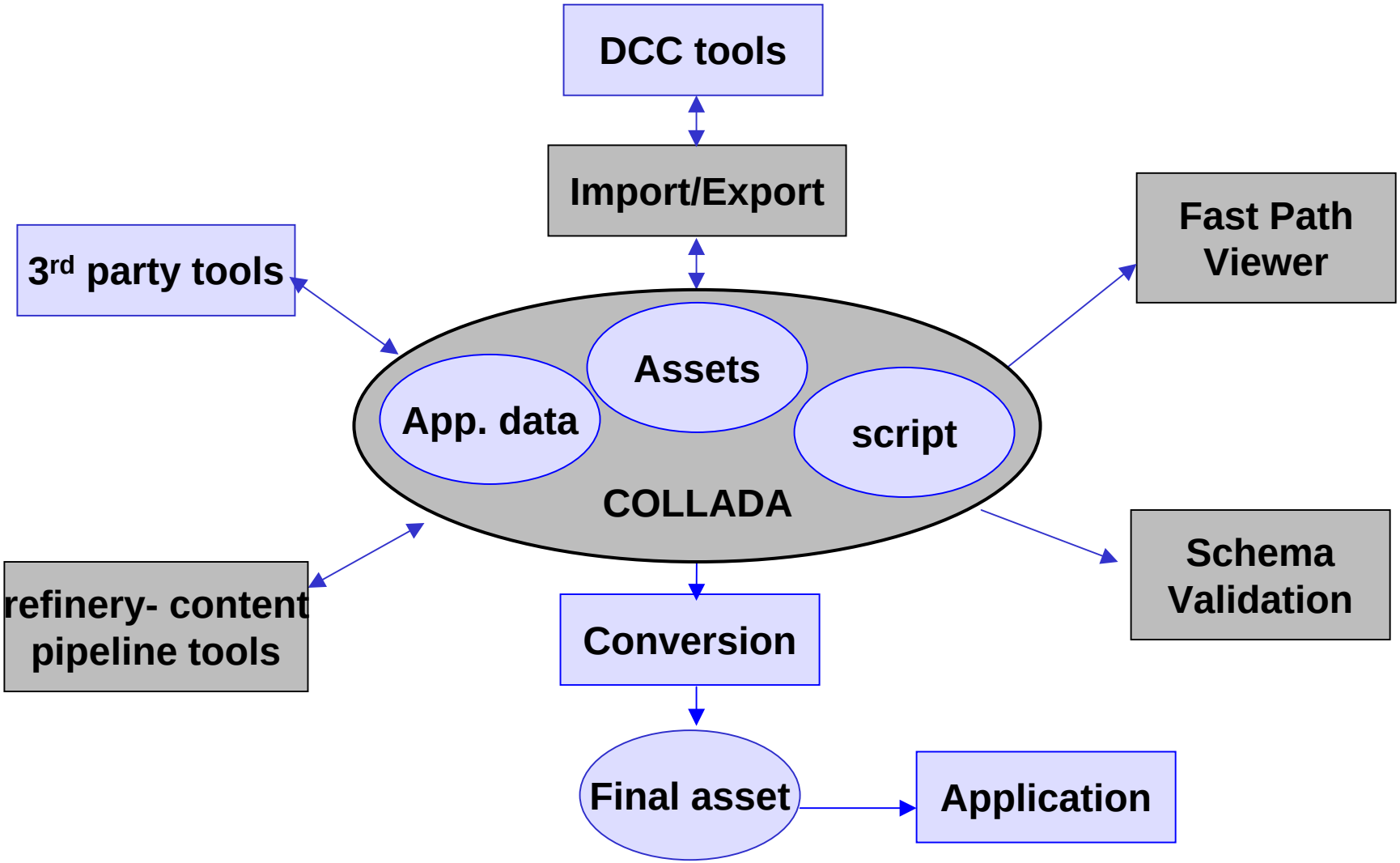
Collada features

- human readable xml format
- a tool-, target- and workflow-independent
- high portability
- higher complexity than all previously discussed formats
- a good place to find Collada models:

  http://sketchup.google.com/3dwarehouse/

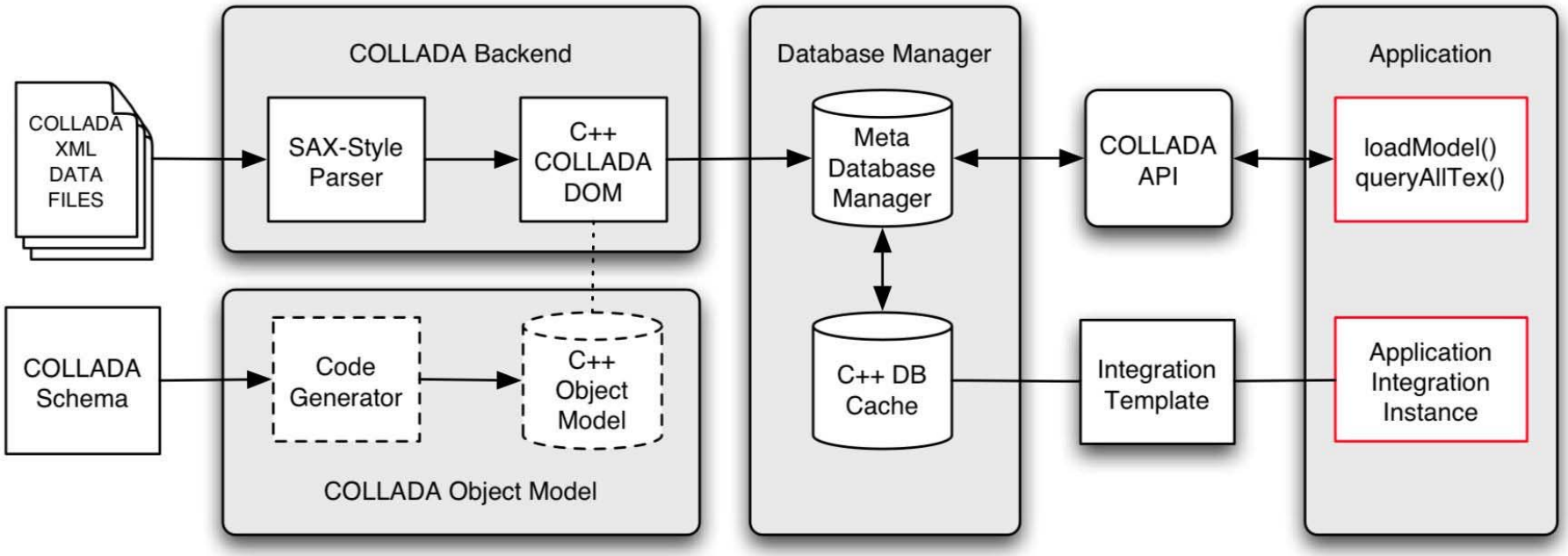
  http://www.daz3d.com/i/3d-models

Collada model



gray parts provided by Collada

Collada API architecture



Collada component features (1)

- **core**
  - mesh geometry
  - vertex skinning
  - morphing
  - animation
  - **assets**
    - a set of information (metadata) that is organized into a distinct collection and managed as a unit
  - **data validation**

Collada component features cont'd (2)

- **Collada FX**
  - is the first cross-platform standard shader and effects definition written in **XML**
  - flexible abstraction for describing material properties across many platforms and APIs
  - offers:
    - abstract material definition
    - effect parameterization and metadata
    - binding to the scene graph
    - inline and external source code or binary

Collada component features cont'd (3)

- **Collada FX**
  - **profiles**
    - `profile_common` for basic interchange between DCC tools
    - `profile_CG` for OpenGL and NVIDIA's Cg Shading Language
    - `profile_GLSL` for OpenGL and the OpenGL Shading Language
    - `profile_GLES` for OpenGL ES 1.0 and 1.1

Collada component features cont'd (4)

- **Collada physix**
  - rigid body dynamics
  - rag dolls
  - constraints
  - collision columes
  - enables data interchange between Ageia (PhysX), Havok, Bullet, ODE and other game physics middleware

Collada open source software (1)

- **Collada DOM**
  - provides a C++ programming interface to load, query, and translate Collada instance data
  - the DOM loads Collada data into a runtime database consisting of structures that mirror those defined in the Collada schema
    - these runtime structures are auto-generated from the current schema, eliminating inconsistency and error

Collada open source software cont'd (2)

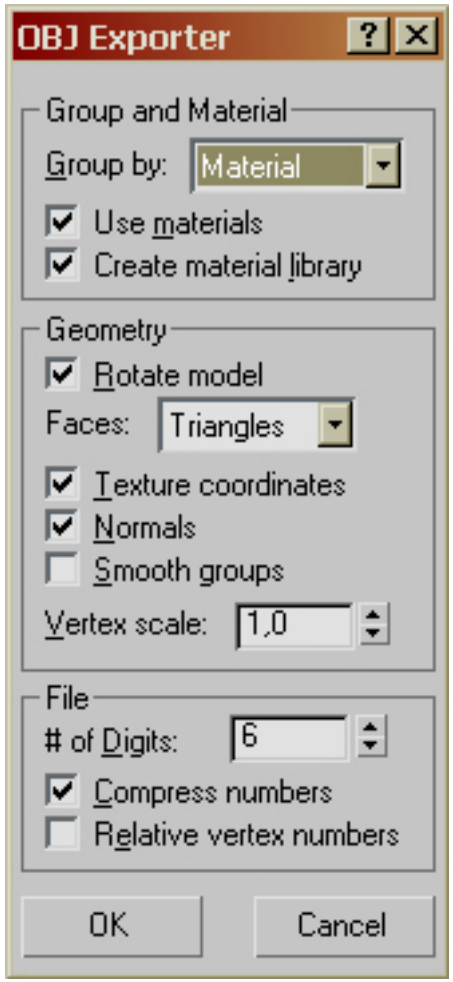
- **FCollada**
  - Open-source C++ library for Collada interoperability.
  - Used by **ColladaMaya**, **ColladaMax**
  - Higher level than COLLADA-DOM
    - Easier to use
    - Hides some of the complexity (can be positive or negative)
- Used by the Feeling Software Viewer

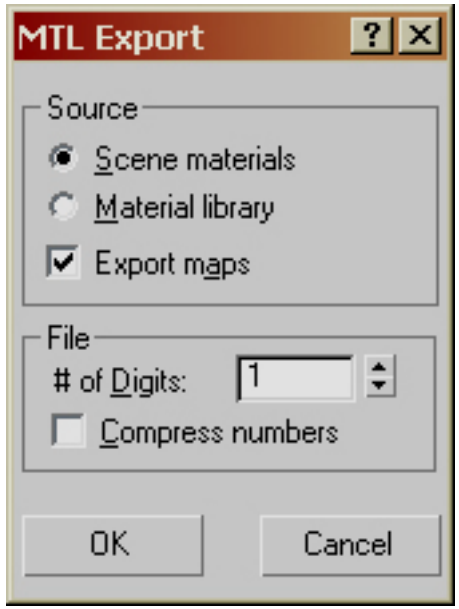
Collada open source software cont'd (3)

- **Refinery – Content Pipeline Tool**
  - what it is:
    - prototype tool chain
    - java-based UI
    - (C++) dll conditioners
    - can run as a batch once conditioning established
  - what it does:
    - triangulate
    - optimize mesh (eliminate T-joints)
    - conversion between Collada versions
    - axis conversion (z_up ->y_up)

Wavefront **obj** - the simplest format (1)

- exported file is in ASCII format





Wavefront **obj** - the simplest format (2)

- **can export**:
  - vertex coordinates
  - vertex texture coordinates
  - vertex normals
- **can not export**:
  - animations
  - scene hierarchies
  - dependencies of any kind
- simple to read and parse (**ASCII format**)
- can handle huge meshes (take **long** for indexing!)
- one has to do all **animation “on foot”**

References (1)

- Quake II

  http://tfc.duke.free.fr/coding/md2-specs-en.html

  http://tfc.duke.free.fr/old/models/md2.htm

  http://cone3d.gamedev.net/cgi-bin/index.pl?page=tutorials/ogladv/tut2

- Doom 3

  http://www.katsbits.com/htm/tools_utilities.htm

  http://tfc.duke.free.fr/coding/md5-specs-en.html

- FBX

http://download.autodesk.com/us/fbx/2010/FBX_SDK_Help/index.html?url=WS1a9193826455f5ff-150b16da11960d83164-6bf0.htm,topicNumber=d0e1370

References (2)

http://usa.autodesk.com/adsk/servlet/pc/item?siteID=123112&id=9245865

- Ogre XML

  http://www.ogre3d.org/wiki/index.php/OGRE_Exporters

  http://www.ogre3d.org/forums/viewtopic.php?f=2&t=55032

  http://www.ogre3d.org/wiki/index.php/LEXIExporter

intermediate format specs:

http://ogre.cvs.sourceforge.net/viewvc/ogre/ogrenew/Tools/XMLConverter/docs/

parser:

http://www.grinninglizard.com/tinyxml/

- Collada

  https://collada.org/mediawiki/index.php/COLLADA_-_Digital_Asset_and_FX_Exchange_Schema

  http://www.khronos.org/files/collada_spec_1_4.pdf

So Far, So Good…

Thank you!