2005:42 HIP

EXAMENSARBETE

Implementation av en COLLADA inläsare för Agency9

Mikael Lagré

Luleå tekniska universitet

Högskoleingenjörsprogrammet
Datorspelutveckling
LTU Skellefteå

2005:42 HIP - ISSN: 1404-5494 - ISRN: LTU-HIP-EX--05/42--SE

Implementation av en COLLADA inläsare för Agency9.

Mikael Lagré
Luleå University of Technology, Sweden
Cellphone: +46(0)733292792
contactme@mikaellagre.com

Förord

Detta examensarbete är gjord för Luleå Tekniska Universitet Institutionen i Skellefteå, våren 2006 hos företaget Agency9, Luleå. Agency9 hade ett behov av att skapa en bättre content pipeline utan att behöva låsa sig fast vid en specifik editor. COLLADA presenterar ett sådant format som möjliggör en lösning. Denna rapport presenterar ett tillvägagångssätt och en lösning för inläsning av formatet COLLADA till AgentFX.

Jag vill tacka Tomas Karlsson på Agency9 som har varit min handledare och hjälpt mig under mitt examensarbete. Jag vill även tacka Khashayar Farmanbar, VD för Agency9 för all hjälp. Jag vill också tacka Johan och Klas, som gjorde deras examensarbete samtidigt som mig, som alltid bidrog med glad stämning på arbetsplatsen.

Abstract

The need for more graphical content in games and real-time applications is increasing with the latest development in hardware. This pushes more work on designers, modelers and animators. There is a need for simplifying the process of moving large data sets from DCC's (Maya, 3dsMax, Blender, etc.), reading that information and present it on screen without using a specific DCC. COLLADA is a way towards this goal. How can COLLADA be implemented in the graphical engine package AgentFX, developed by Agency9? This thesis paper presents a way of using Java and COLLADA with JAXB to implement support for the AgentFX, to display complex scenes build from Maya Unlimited 6.5. I begin this paper by presenting different XML parsers and continue with presenting the interface for the JAXB process and its integration with the overall COLLADA reader design. I discuss pros and cons for the JAXB settings file and how it is important to understand its powerful features. This paper then discusses the implementation solutions I took when adding support for COLLADA in AgentFX. "Piglet and Pretorean" is a complex scene which is presented as a demonstration of my successful work with this project. This paper ends with a discussion for future development of my work together with the future development of the COLLADA format.

Sammanfattning

Behovet av mängden grafiska detaljer i spel och realtidsapplikationer ökar i och med intåget av nyare och kraftfullare hårdvara. Detta sätter mera press på grafiker. Det finns ett behov av att simplifiera processen med att flytta stora datamängder från olika editorer (Maya, 3dsMAX, Blender, mm), processa denna information och presentera det på skärmen utan att låsa sig fast vid en specifik editor. COLLADA är ett steg mot detta mål. Hur kan COLLADA implementeras i grafikmotorn AgentFX, utvecklad av Agency9? Detta examensarbete presenterar ett tillvägagångssätt genom Java och COLLADA med JAXB att implementera stöd för COLLADA i AgentFX, för att rendera komplexa scener byggda i Maya Unlimited 6.5. Jag börjar denna rapport med att presentera olika sätt att läsa in XML dokument och fortsätter presentera gränssnittet med JAXB processen och integrationen i min COLLADA inläsare. Jag diskuterar fördelar och nackdelar med egenskapsfilen i JAXB och hur det är viktigt att förstå dess inverkan i designen. Denna rapport diskuterar vidare en lösning för implementation av omvandling från COLLADA formatet till AgentFX. "Piglet and Pretorean" är en komplex scen vilket presenteras som en demonstration över mitt lyckade resultat i arbetet. Denna rapport avslutar med en diskussion över framtida möjligheter och utsikter av mitt arbete samt den framtida utvecklingen av COLLADA formatet.

Innehållsförteckning

1 Introduktion

1.1 Bakgrund

Dagens spelutveckling håller på att bli större i skala med mindre utvecklingstid och med mera pengar involverade än någonsin förr. I och med lanseringen av nästa generations konsoler kommer det att finnas ett större behov av grafiker för uppbyggnad av karaktärer och miljöer inom spel. Kostnaden för att anställa och utbilda grafiker inom ett visst verktyg för skapandet av dess kreationer kan bli mycket hög. Det finns redan idag ett behov från spelföretagen av att separera data från modelleringsverktygen och strukturen som används inom implementationen av spelen. Det finns ett behov av att programmerare inte behöver skapa stöd för olika format, vilket tar tid och har hög kostnad, och grafiker inte behöver lära sig ett specifikt verktyg utan tillåts frihet att jobba i sitt eget modelleringsprogram.

1.1.1 COLLADA

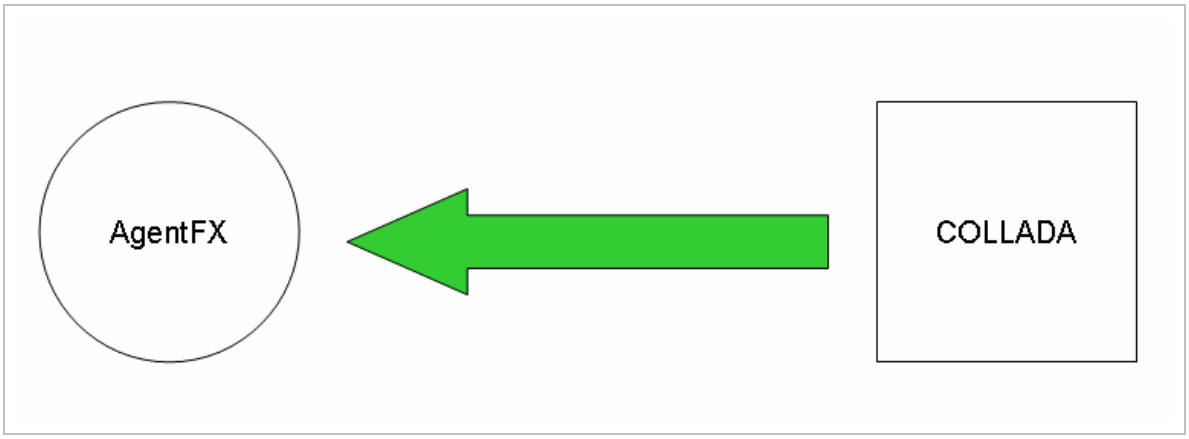
COLLADA (COLLAborative Design Activity) är ett format baserat på XML [1] som sparar renderingsordning och data från olika 3D-modelleringsprogram (Maya, 3D Studio Max). COLLADA är utvecklad av ”SCEA R&D group” och är ett open-source projekt under tillväxt [2]. Se bilaga A för XML schema uppbyggnaden för COLLADA samt bilaga C för ett exempel på en collada struktur.

1.1.2 AgentFX

AgentFX [12] är ett API skriven i Java för rendering av grafik utvecklad av Agency9. AgentFX med tillhörande renderingspaket och API har stöd för bland annat animerade karaktärer, skuggvolymer, multi-texturering, cg-shader med mera.

1.1.3 Problemområde

Mitt problemområde är hur man skapar en bra implementation och designstruktur så att AgentFX kan ha support for COLLADA formatet. Hur skapar jag bäst en implementation och design för framtida versioner av COLLADA och AgentFX? Och hur skapar jag en struktur som stämmer bra överens med AgentFX och tillhörande paket.



1.1.4 Syfte

Examensarbetet syfte var för min egen del att lära mig ett nytt format och hur jag kan gå från detta nya format till en renderingsstruktur i en grafikmotor. Jag ville utöka mina kunskaper inom grafikernas arbetsmetoder och verktyg samt min egen förståelse för hur processen från grafikernas arbete till programmerarnas arbete fungerar. En stor del av syftet med examensarbetet var för mig att kunna tillämpa mina tidigare kunskaper på detta nya okända område och lyckas prestera ett resultat som var användbart för företaget.

1.1.5 Begränsningar

Inläsaren ska vara skriven i Java och fungera på de plattformar som AgentFX stödjer samt läsa in de COLLADA dokumentet som har genererats utifrån en Maya Unlimited 6.5 exporter.

2 Teori

2.1 XML inläsare

De mest vanliga verktygen och metoderna inom Java för att läsa av ett XML schema och ett XML dokument presenteras här.

2.1.1 SAX

SAX (Simple API for XML) är ett händelsebaserat API ursprungligen skapat av Davig Meggison och är nu släppt för allmänheten [9]. SAX använder ett callback gränssnitt för att returnera XML data till klientapplikationen. Detta gör att SAX är snabbt och hanterar minne effektivt eftersom det inte behöver spara hela dokumentet i minnet. Att använda SAX kan ta sin tid i designprocessen eftersom man behöver designa upp sin egen struktur för lagring av elementen i XML dokumentet [8].

2.1.2 DOM

DOM (Document Object Model) processar XML dokumentet och skapar en trädliknande struktur över objekten i dokumentet. API strukturen i DOM är väldigt komplex. DOM kan, till skillnad från SAX, också skapa XML dokument från objekt och data i dess trädstruktur. På grund av lagringsutrymmet av dokumentets skapade trädstruktur är inte DOM så minneseffektivt och passar heller inte bra för applikationer som använder en ström av data för inläsning [11].

2.1.3 JAXP

JAXP [3] är Sun Microsystems egna Java API för XML inläsning och binder tillsammans SAX och DOM i java version 1.4 och senare [6]. JAXP, tillsammans med dess tillägg av API för skapande av objekt, binder SAX och DOM i ett paket men vid implementationsprocessen måste man ändå välja mellan SAX eller DOM för inläsning av XML dokumentet.

2.1.4 JDOM

JDOM [5] försöker att ta bort den svåra hårdkodningen från DOM API:et och möjliggör genom JDOM's API en lättare lösning för att läsa av ett XML dokument. Precis som DOM så skapas en trädliknande struktur som sparas i minnet. JDOM är mera effektivt på enkla dokument och skapar konkreta klasser och gränssnitt för att modellera sitt trädgränssnitt. JDOM passar bra när XML Schema strukturen är bra definierad men den kan ha problem med godtyckliga XML strukturer.

2.1.5 JAXB

JAXB [3, 13] använder sig av en kompilerare för att binda XML scheman eller DTD's till konkreta klasser i Java. De kompilerade klasserna används precis som i JDOM för att bygga upp en trädstruktur över XML dokumentet och möjliggör direkt manipulation av XML elementens data. JAXB paketet innehåller både kompilerare för att kompilera fram schema klasser samt en inläsare för att läsa av XML dokumentet. Den innehåller dessutom funktionalitet för att skapa XML dokument utifrån ett redan skapat träd. Denna

process, att bygga trädliknande strukturer utifrån konkreta klasser baserade på XML strukturen, kan vara väldigt användbar för vissa XML scheman men oftast om schemat har en väldigt specificerat syntax. [7] tar upp en diskussion om data bindning utifrån XML schema specifikationer.

2.1.6 Andra metoder

Några andra lösningar för att läsa in XML dokument är t.ex. dom4j (baserat på JDOM), ElectricXML (bra för små applikationer) och XMLPull.

3 Design

Efter att ha titta på fördelar och nackdelar med olika XML inläsare i Java så valde jag relativt tidigt att använda mig av JAXB för att läsa in COLLADA formatet. JAXB passar utmärkt för inläsning eftersom COLLADA schemat har en väldigt specifik syntax. JAXB tillåter mig kompilera fram klasser utifrån COLLADA schemat, bygga en trädstruktur utifrån XML dokumentet, samt att det tillhandahåller ett enkelt gränssnitt för åtkomst av dokumentets element och data i denna trädstruktur (se fig. 1).

Abstrakt design över processen

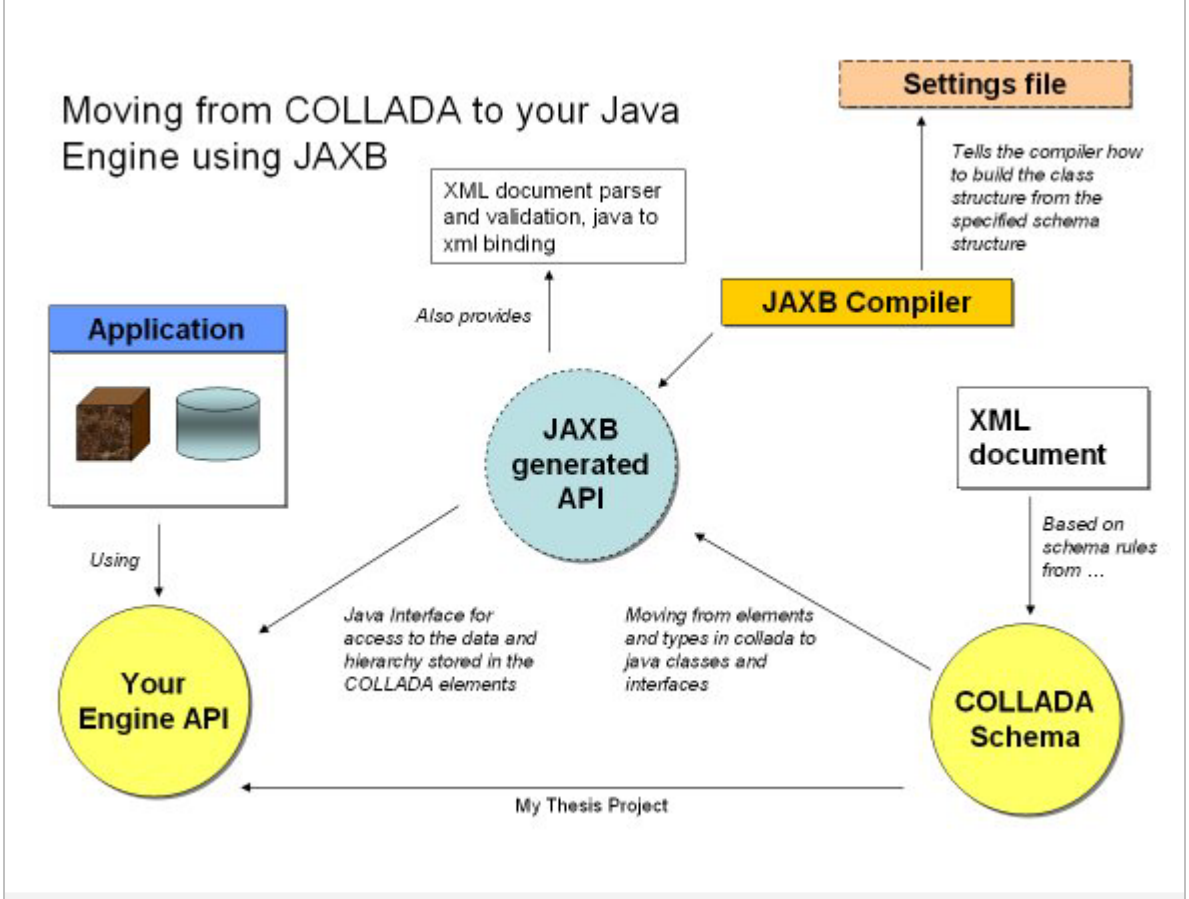


Fig. 1. Att gå från COLLADA till ett JAVA gränssnitt genom JAXB

3.1 Klassdesign med JAXB

JAXB använder sig av en kompilerare kallad XJC (XML Java Compiler) för att kompilera fram klasser ur ett XML Schema [14]. Denna kompilator använder sig av ett XML dokument, med schema regler enligt W3 och JAXB specifikation som innehåller egenskaper för hur kompileringen från ett XML schema till Java klasser ska ske. Specifikationen för JAXB, XJC samt egenskapsfilen kan laddas ner på följande URL: http://java.sun.com/xml/downloads/jaxb.html

3.1.1 Egenskapsfilen

Egenskapsfilen, XML dokumentet för kompileringsprocessen, består i stort av ett antal element som definierar hur klasser ska skapas, vilka som ska skapas och vilka java typer som ska skapas beroende på typer inom schema layouten. Filen är i detta fall extern från schema strukturen men kan även skrivas direkt i schemat som ska kompileras. För COLLADA har jag valt att skapa en extern fil eftersom schema layouten kan komma att ändras i fortsättningen. En extern fil tillåter mig då att enkelt kunna lägga till alternativ för nya element som ska kompileras enligt en ny specifikation på COLLADA.
Här nedan presenteras de viktigaste punkterna för uppbyggnaden av egenskapsfilen som jag skrev för att kunna kompilera fram gränssnittet mellan COLLADA och AgentFX. Se bilaga B för hela XML dokumentets uppbyggnad.

3.1.2 Globala egenskaper

Egenskapsfilen definierar olika globala egenskaper vilka gäller som mall för varje element i XML schemat samt för kompileringsprocessen som helhet.

Globala egenskaper

```
<jxb:globalBindings
    fixedAttributeAsConstantProperty="false"
    collectionType="java.util.ArrayList"
    typesafeEnumBase="xs:NMTOKEN"
    choiceContentProperty="true"
    typesafeEnumMemberName="generateError"
    bindingStyle="elementBinding"
    enableFailFastCheck="false"
    generateIsSetMethod="false"
    underscoreBinding="asWordSeparator">

  <!-- Global java type bindings -->
  <jxb:javaType name="int" xmlType="xs:nonNegativeInteger"
    parseMethod="javax.xml.bind.DatatypeConverter.parseInt"
    printMethod="javax.xml.bind.DatatypeConverter.printInt"/>

</jxb:globalBindings>
```

fig. 2. Globala inställningar i egenskapsfilen för klasskompilering med XJC.

Den största och viktigaste inställningen i egenskapsfilen är huruvida man ska använda sig av elementBinding eller modelGroupBinding för att skapa klasstrukturer. Standardinställningen är elementBinding vilket skapar ett enklare och mera lättillgängligt interface för att hämta information per element än objectModelBinding. ElementBinding tillsammans med unika egenskapsinställningar per element visade sig vara tillräckligt för att producera fram en bra designstruktur.

Genom att ange xs:TOKEN till typesafeEnumMemberName inställningen skapar XJC enumklasser för de element som har typen TOKEN. Som standardinställning skapar XJC enumklasser utav xs:NAME element vilket inte passar lika bra för COLLADA formatet.

Jag har även valt att konvertera XML typen nonNegativeInteger till javatypen int. Detta beror på att grundinställningen för XJC konverterar nonNegativeInteger typen till java.lang.BigInteger vilket är en onödigt stor precision i praktiken. Att låta XJC direkt konvertera till javatypen int sparar tid vid implementationen eftersom jag då inte behöver konvertera till javatypen int internt i kod.

3.1.3 Schema egenskaper

Egenskaper som gäller för hela COLLADA schemat ligger under schema egenskaper. Detta innebär bland annat definition för paketstrukturen samt prefix och suffix på genererade klassnamn.

Schema egenskaper

```
<jxb:schemaBindings>
  <jxb:package name="com.agency9.agentfx.io.collada">
    <jxb:javadoc><![CDATA[<body>Package level
        documentation for generated package com.agency9.agentfx.io.collada</body>]]>
    </jxb:javadoc>
  </jxb:package>
  <jxb:nameXmlTransform>
    <jxb:elementName prefix="A9COLLADA" suffix="Element"/>
    <jxb:typeName prefix="A9COLLADA" suffix="Constants"/>
    <jxb:anonymousTypeName prefix="A9COLLADA" suffix="Type"/>
  </jxb:nameXmlTransform>
</jxb:schemaBindings>
```

fig. 3. Egenskaper för hela schema layouten i COLLADA.

Jag följer här Agency9 och AgentFX's kodningsstandard och lägger klasser för inläsare i paketet ”com.agency9.agentfx.io”. Jag har även valt att lägga till prefixet ”A9COLLADA” för alla genererade klasser, suffixen ”Type” för gränssnittsklasser samt suffixen ”Element” för konkreta subklasser. Denna namngivning är satt så att den ska passa ihop med övrig namngivning i samma paket samt i AgentFX som helhet.

3.1.4 Typomvandling

Från egenskapsfilen kan man ange omvandlingsprocedurer från en COLLADA XML-typ till en javatyp. Denna oerhörda flexibilitet gör att man redan vid kompileringsstadiet kan bestämma vilken information som ska kunna hämtas från olika element. Nedan följer ett exempel på typomvandlingssyntax och hur man enkelt kan definiera omvandling från typer i COLLADA till typer i AgentFX.

COLLADA Schema	Egenskapsfilen
<xs:simpleType name="ListOfFloats"> <xs:list itemType="xs:double" /> </xs:simpleType>	<jxb:bindings node="//xs:simpleType[@name='ListOfFloats']"> <jxb:javaType name="com.agency9.buffer.A9FloatBuffer" parseMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseFloatBuffer" printMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseString"/> </jxb:bindings>

Fig. 4. COLLADA typen ListOfFloats omvandlas till en A9FloatBuffer i egenskapsfilen.

En typomvandling kräver en referens till två funktioner. En funktion som tar in en strängparameter och omvandlar parametern till en javatyp, samt en som tar in en javatyp och omvandlar till en sträng (java.lang.String). Funktionerna måste vara skrivna redan vid kompileringsstadiet eftersom XJC använder funktionerna internt för att bygga upp klasser.

Om ingen typomvandling hade definierats i en egenskapsfil skulle typen ListOfFloats ha omvandlats till javatypen java.util.ArrayList (se fig. 2) vilket hade försvårat implementationen. I och med att det redan finns en bättre typ i AgentFX paketet är det fördelaktigt att använda sig av den istället.

Notera skillnaden mellan den globala inställningen över att omvandla typen xs:nonNegativeInteger till javatypen int (se fig. 2) och omvandlingsinställningen för en xs:simpleType. Den globala inställningen gäller en W3 definierad XML typ medan schema inställningen gäller en typ inom COLLADA schemat (se fig. 4).

Värt att notera här är att COLLADA definierar typen i ListOfFloats som xs:double. Eftersom AgentFX endast jobbar med precisionen i javatypen float har jag valt att använda A9FloatBuffer för just denna typomvandling.

3.1.5 Enumerationsklassers definition

Egenskapsfilen tillåter oss även att bygga upp enumerationsklasser från COLLADA till genererade klasser med kompileraren XJC. Genom att ange inställningen xs:NMTOKEN i de globala inställningarna (se fig. 2) kommer de COLLADA element som är definierade

som en samling av xs:NMTOKEN att bli konstanter i en enumerationsklass i java. Detta åskådliggörs lättare med ett exempel från syntaxen i COLLADA schemat och motsvarande syntax i egenskapsfilen.

COLLADA Schema

```
<xs:simpleType name="LibraryType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="ANIMATION" />
      <xs:enumeration value="CAMERA" />
      <xs:enumeration value="CODE" />
      <xs:enumeration value="CONTROLLER" />
      <xs:enumeration value="GEOMETRY" />
      <xs:enumeration value="IMAGE" />
      <xs:enumeration value="LIGHT" />
      <xs:enumeration value="MATERIAL" />
      <xs:enumeration value="PROGRAM" />
      <xs:enumeration value="TEXTURE" />
    </xs:restriction>
</xs:simpleType>
```

Egenskapsfilen

```
    <jxb:bindings
node="//xs:simpleType[@name='LibraryType']">
      <jxb:typesafeEnumClass
name="A9COLLADALibraryConstants">
      </jxb:typesafeEnumClass>
    </jxb:bindings>
```

Fig. 5. COLLADA typer omvandlas till enumerationsinställningar i egenskapsfilen.

3.1.6 Lokala egenskapsinställningar

Varje element i COLLADA kan ha specialiserade egenskaper definierade i egenskapsfilen för att skapa en effektivare implementation och användning utav det genererade gränssnittet. Jag valde att skriva egenskaper för varje element och typ definierad i COLLADA schemat. Detta beror på möjligheten att lättare kunna bygga vidare på en anpassad klasstruktur för nyare specifikationer av COLLADA schemat. Generellt har de lokala egenskapsinställningarna följt riktlinjer så att ett klassnamn har skrivits för varje element, ett funktionsnamn och en egenskap har genererats för varje typ som är möjlig att generera. I de fall där en samling objekt inte kan delas upp i olika privata egenskaper hos ett element har jag valt att skapa funktionsanrop som är logiska för varje element. Nedan följer två exempel på lokala egenskapsinställningar för COLLADA element tillsammans med en förklaring över valda egenskaper för respektive element.

COLLADA Schema element	Egenskapsinställningar
<xs:element name="library"> <xs:complexType> <xs:choice minOccurs="0" maxOccurs="unbounded"> <xs:element ref="animation" /> <xs:element ref="camera" /> <xs:element ref="code" /> <xs:element ref="controller" /> <xs:element ref="geometry" /> <xs:element ref="image" /> <xs:element ref="light" /> <xs:element ref="material" /> <xs:element ref="program" /> <xs:element ref="texture" /> </xs:choice> <xs:attribute name="id" type="xs:ID" /> <xs:attribute name="name" type="xs:NCName" /> <xs:attribute name="type" type="LibraryType" use="required" /> </xs:complexType> </xs:element>	<jxb:bindings node="//xs:element[@name='library']"> <jxb:class name="A9COLLADALibraryElement"/> <jxb:bindings node=".//xs:complexType"> <jxb:bindings node=".//xs:choice"> <jxb:property name="LibraryElements"/> </jxb:bindings> </jxb:bindings> </jxb:bindings>

Fig. 6. COLLADA element och deras respektive lokala egenskapsinställningar.

De inställningar jag gör för libraryelementet i egenskapsfilen är begränsade enligt JAXB specifikationen för en egenskapsfil. Jag väljer att kalla den generade konkreta klassen för A9COLLADALibraryElement. Gränssnittstypen kommer (enligt fig.3) att bli A9COLLADALibraryType. Anledningen till att jag har valt att för de flesta element namnge den konkreta klassen är för att det genererade klassnamnet kommer utan lokala klassnamninställningar att börja med liten bokstav efter prefixnamnet A9COLLADA. A9COLLADALibaryElement skulle utan en lokal namninställning således bli A9COLLADAlibraryElement. För att inte strida mot rådande kodningsstandard inom java kulturen och för att få en renare API struktur valde jag att överlagra klassnamnsinställningarna för de flesta elementen.

I och med att libraryelementet i COLLADA schemat definierar upp en lista, med element referenser, som kan vara oändligt stor kommer XJC att skapa en ArrayList (enligt fig.2) över möjliga element i listan. Genom egenskapsfilen kan jag välja att skapa funktionsnamnet för hämtning av de element som ingår i denna lista. I detta fall kallar jag den för libraryElements. XJC kommer att lägga till funktioner för att hämta lista och sätta listan med prefixen set respektive get. Funktionsnamnet för att hämta listan i A9COLLADALibraryType kommer då att bli getLibraryElements. Dessa små ändringar tyckte jag var nödvändiga för att få en så pass ren och anpassad API struktur att jobba med vid implementationsfasen.

Ytterliggare ett exempel på lokala egenskapsinställningar i egenskapsfilen:

COLLADA Schema element	Egenskapsinställningar
<xs:element name="geometry"> <xs:complexType> <xs:sequence> <xs:element ref="mesh" minOccurs="1" maxOccurs="1" /> <xs:element ref="extra" minOccurs="0" maxOccurs="unbounded" /> </xs:sequence> <xs:attribute name="id" type="xs:ID" /> <xs:attribute name="name" type="xs:NCName" /> </xs:complexType> </xs:element>	<jxb:bindings node="//xs:element[@name='geometry']"> <jxb:class name="A9COLLADAGeometryElement"/> <jxb:bindings node=".//xs:complexType"> <jxb:bindings node=".//xs:sequence"> <jxb:bindings node=".//xs:element[@ref='mesh']"> <jxb:property name="Mesh"/> </jxb:bindings> <jxb:bindings node=".//xs:element[@ref='extra']"> <jxb:property name="Extra"/> </jxb:bindings> </jxb:bindings> </jxb:bindings> </jxb:bindings>

Fig. 7. COLLADA element och deras respektive lokala egenskapsinställningar.

I detta exempel har jag möjlighet att dela upp geometrielementets två element referenser, ”mesh” och ”extra”, som två separata egenskaper inom det genererade gränssnittet A9COLLADAGeometryType. Detta beror på att de ingår i en ”xs:sequence” och måste således komma i ordning. Detta kan jag utnyttja och skapar inställningar för två funktioner för att hämta data, ”getMesh” samt ”getExtra”. I en standardkompilering hade en klass, Geometry, samt ett gränssnitt, GeometryType, skapats där gränssnittet hade fått funktionen getMeshOrExtra. I och med min överlagring i egenskapsfilen kommer jag nu få en bättre struktur på funktionsnamn och klassnamn när arbetet med typomvandling och JAXB processen ska implementeras.

3.2 Gränssnitt

Nedan presenteras de gränssnitt som ingår i den färdiga designen. Processen för att generera Java klasser för COLLADA strukturen via XJC och JAXB behandlades i avsnitt 3.1. Den övriga designen, som tas upp i detta avsnitt, handlar om funktionaliteten i gränssnitten Reader, User Interface, Agent FX samt Application. Dessa gränssnitt, deras innebörders ordning, samt relationerna mellan andra gränssnitt tas upp här.

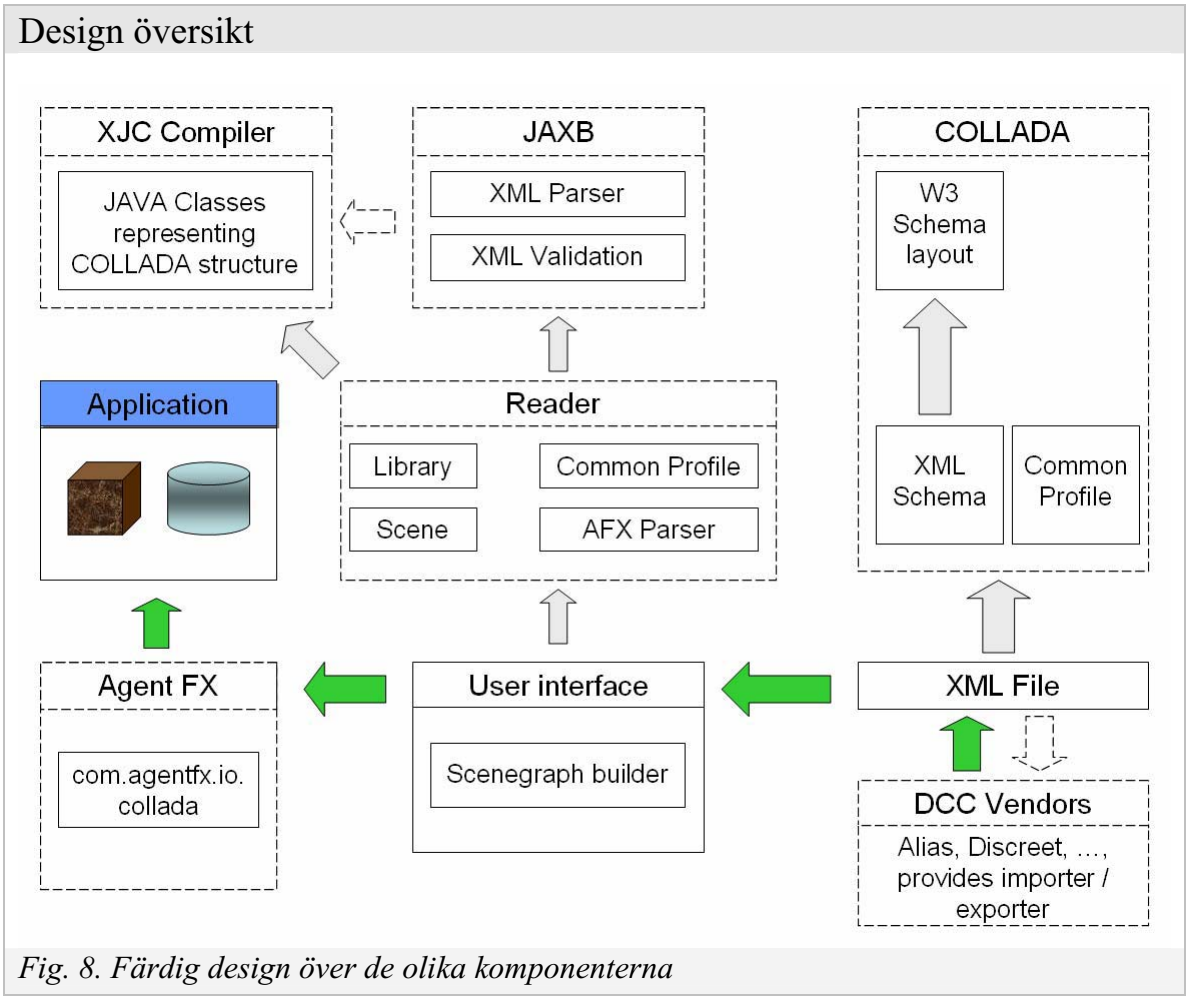


Fig. 8. Färdig design över de olika komponenterna

3.2.1 User Interface

Användargränssnittet består utifrån sett endast av en funktionalitet. Att från en COLLADA fil omvandla informationen till en A9Node. Noden innehåller hela renderingsträdet för att kunna presenteras korrekt i en Application som använder sig av AgentFX. Användargränssnittet är således väldigt begränsat. Detta beror på flera orsaker men den främsta är att COLLADA samt inläsaren (se avsnitt 3.2.2) tillåter större funktionalitet och uppbyggnad direkt i modelleringsprogrammet. Därför har jag valt att

lägga större vikt på kontrollen vid den grafiska uppbyggnaden av COLLADA formatet istället för programmeraren vid implementation. En annan stor anledning är att kontrollen för inläsningen kan ske efter omvandlingen till renderingsträdet och inte under inläsning. Detta gör att möjliga objekt eller inbördes ordning i renderingen kan ändras med AgentFX funktionalitet istället för systemet som används internt utav Reader.

3.2.2 Reader

Reader (inläsare) är modulen som User Interface gränssnittet är uppbyggd kring. Reader-gränssnittet är i sin tur uppdelad i mindre moduler. Reader använder sig av JAXB gränssnittet för validering och för unmarshallprocessen (läs mer om det under avsnitt 4.3.1). De två stora modulerna i Reader är Library samt Scene. COLLADA schemat är uppbyggd på ett liknande sätt med flera library element och ett scene element vilket gjorde det naturligt att skapa dess två moduler. Det möjliggör också att jag kan processa så mycket information som jag har tillgänglig i Library för att snabba upp inläsningen i Scene gränssnittet.

3.2.2.1 Library

Library gränssnittet i Reader använder sig av AFX Parser för att omvandla JAXB genererade klasser till AFX klasser. Library är således en inläsare för data i COLLADA. För varje genererad JAXB klass (som utgår ifrån ett COLLADA element i COLLADA schemat) finns en motsvarade funktionalitet som Library använder sig av för att processa denna klass till data som kan användas för att bygga upp renderingsinformation i AgentFX.

3.2.2.2 AFX Parser

AFX Parser har hand om omvandlingen från COLLADA typer och JAXB genererade typer till AgentFX klasser. Framförallt är den största funktionen för AFX Parser att omvandla information i de olika library elementen i COLLADA till AgentFX objekt så att dessa objekt senare kan användas som referens vid inläsningen av scene elementet.

3.2.2.3 Scene

Scene modulen behandlar scene elementet i COLLADA formatet och kopplar ihop data information i Library med logisk renderingsordning och uppbyggnad utav noder.

3.2.2.3 Common Profile

Common Profile är en samling konstanter för COLLADA's Common Profile enligt specifikationen i COLLADA 1.3.0. Förutom dessa finns även jämförelsefunktioner mellan strängar och konstanter.

4 Implementation

4.1 Överblick

I och med att jag använde mig av JAXB genererade klasser förenklades implementationen något avsevärt. Jag kan enkelt hämta element ur ett COLLADA dokument och direkt läsa av värden i elementet via anpassade funktioner definierade i egenskapsfilen.

4.1.1 Process

Processen för omvandling från COLLADA till ett AgentFX renderingsträd börjar med ett anrop till en statisk A9COLLADAReader funktion som tar in en textsträng som parameter. Denna parameter anger vilket COLLADA dokument som ska läsas in. Detta dokument öppnas och används sedan av JAXB för sin Unmarshallingprocess (bygger upp sin interna trädstruktur utav innehållet i dokumentet).
Efter den processen stegar jag igenom denna trädstruktur och läser först in trädets olika Library element via funktionalitet i A9COLLADALibrary.
I A9COLLADALibrary omvandlar jag informationen till typer så nära AgentFX som möjligt och lagrar objekten i en HashMap. Detta är mycket effektivt och säkert eftersom COLLADA specifikationen kräver ett unikt ID för varje element i hela dokumentet vilket passar perfekt för att lagra de omvandlade objekten i en HashMap med deras ID som nyckel. I A9COLLADAScene läser jag sedan av trädstrukturens scene element som består av nodhierarki. Denna nodhierarki läses av och anpassas beroende på dess information till A9Node objekt i ett renderingsträd. A9COLLADAScene hämtar data ur library, och kopierar data i de fall det är nödvändigt, för varje nod i nodhiearkin i scene elementet. Resultatet vid inläsningen i A9COLLADAScene läggs sedan så att en A9BufferNode och en A9Camera alltid kommer att ligga som föräldrar till renderingsordningen som lästes in i A9COLLADAScene. Detta beror på hierarkin som AgentFX använder där en A9BufferNode behöver ligga överst i de flesta renderingar tillsammans med en A9Camera som barn för att kunna se det som har renderats.

4.1.2 Kodningsstandard och syntax

Jag har följt kodningstandard enligt [15]. Jag har valt att göra min syntax så att en utbyggnad utav koden ska vara enkelt och lätt. Klassnamn och funktionsnamn är valda så att de stämmer bra överens med kodningsstandarden och syntaxen i klasser i övriga delar utav AgentFX paketet.

4.2 Klasser

Klasser, deras funktionalitet och struktur, i paketet com.agency9.agentfx.io.collada samt några klasser ur paketet com.agency9.agentfx.skeleton som är relaterade till arbetet med inläsaren diskuteras här.

4.2.1 A9COLLADAReader

A9COLLADAReader tillämpar User Interface gränssnittet samt Reader gränssnittet. Klassen består av en statisk funktion som genom JAXB läser in en java resource. Denna resource, till exempel en lokal fil, kommer genom den så kallade unmarshallprocessen omvandla XML dokumentet till JAXB's interna träd och ge som resultat topnoden för trädhierarkin. Denna topnod är av typen A9COLLADACOLLADAElement eftersom denna är definierad i COLLADA specifikationen som det översta elementet i schemat. JAXB tillåter även validering av XML dokumentet gentemot schemat. Denna valmöjlighet är i A9COLLADAReader avstängd på grund av två orsaker. Dels är denna validering relativt långsam vilket inte är att föredra i en realtids applikation, dels är valideringen onödig eftersom man från användare av klassen kan anta att de dokument som läsas in är genererade och därmed "well-defined" enligt W3 och COLLADA schemat. A9COLLADAReader innehåller dessutom en referens till en A9COLLADALibrary och en A9COLLADAScene. A9COLLADALibrary objektet anropas efter unmarshallingprocessen för att omvandla lagrad data i Library elementen i COLLADA dokumentet och A9COLLADAScene anropas efter omvandlingen i A9COLLADALibrary för att bygga upp renderingsträdet. Funktionen returnerar en A9Node som innehåller all information för att skapa ett resultat på skärmen.

4.2.2 A9COLLADALibrary

A9COLLADALibrary tillämpar Library gränssnittet. Denna klass har hand om inläsningen av de olika bibliotekstyper som finns inom COLLADA dokumentet samt att spara information från de olika bibliotekstyperna. A9COLLADALibrary är således ett stort bibliotek som sparas information från de olika biblioteksselementen som existerar. Ett A9COLLADACOLLADAElement granskas och möjliga bibliotek hämtas från elementet. Beroende på typen av biblioteket (geometri, animation, kamera mm) anropas olika funktioner i A9COLLADAParser som returnerar ett objekt som lagras i en hashmap i A9COLLADALibrary. I några enstaka fall returnerar inte A9COLLADAParser objekt eftersom en direkt omvandling från en COLLADA typ till en specifik typ inte är nödvändig eller ens möjlig. I de fall skickas oftast biblioteksinformation vidare till omvandlingsklassen som då har möjlighet att själv lagra objekt i listan. A9COLLADALibrary har även funktionalitet för att lagra objekt i sitt bibliotek samt att hämta information ur det via en nyckel som parameter.

4.2.3 A9COLLADAScene

A9COLLADAScene tillämpar Scene gränssnittet. Klassen innehåller all funktionalitet för att bygga den logiska renderingsordningen som påträffas i COLLADA dokumentet samt hur anpassningen sker till strukturen i AgentFX renderingsträd. Två rekursiva funktioner finns i A9COLLADAScene. En bygger upp AgentFX nod struktur till ett träd. En annan bygger upp ett A9Skeleton med hjälp av den eventuella jointhierarkin som finns i

COLLADA dokumentet. A9COLLADAScene använder sig av den information som nu är lagrad i A9COLLADALibrary.

4.2.3 A9COLLADAParser

A9COLLADAParser tillämpar AFXParser gränssnittet och tillhandahåller således ett gränssnitt för alla sorters typomvandlingar. Förutom omvandlingsfunktioner från COLLADA element till AgentFX typer så finns även omvandling från strängar till flyttal, matriser och andra typer. Dessa omvandlingsfunktioner används redan av egenskapsfilen vid JAXB kompileringen (se avsnitt 3.1.4). Klassen innehåller dessutom hjälpfunktioner för att hämta ut den så kallade ”Common Technique” ur ett COLLADA element [2]. En rekursiv funktion för att kopiera nodinformation finns även här som kopierar sig själv och sina barn men delar på komponenter funna i noden.

4.2.4 A9COLLADACommonProfile

COLLADA 1.3.0 specifikationen definierar en ”Common Profile” som består av konstanter som inte är specificerade i COLLADA schemat. Klassen A9COLLADACommonProfile tillämpar Common Profile gränssnittet och är en klass som i sin tur har en samling av klasser innehållande strängkonstanter och enumerationsobjekt som alla ingår i ”Common Profile”. A9COLLADACommonProfile har funktionalitet för att konvertera en sträng till javatypen int. Klassen existerar för att skapa en samlingsplats för den COLLADA specifikation som inte ingår direkt i COLLADA schemat och möjliggör en säkrare lättare syntax vid implementationen utav de övriga gränssnitten.

4.2.5 A9COLLADAKeyFrame

En A9COLLADAKeyFrame definierar upp en tid och ett tillhörande värde bestående av ett godtyckligt objekt. I vissa avseenden i inläsningen är inte key-frame animationen definierad förens i andra element varav det behövdes en mellanlagring mellan ren data och logiken i en animation. A9COLLADAKeyFrame har funktionalitet för att hämta objektet, samt sätta objektet samt sätta tiden och hämta tiden.

4.2.6 A9COLLADAKeyFrameList

A9COLLADAKeyFrameList är en samling A9COLLADAKeyFrames sparade i en lista. Klassen har funktionalitet för att lägga till key-frames samt hämta key-frames via indexeringsparameter.

4.2.7 A9COLLADAAddressSyntax

COLLADA specifikationen definierar upp en Address Syntax som en sträng bestående av en ID från instansdokumentet följt av en eller flera subidentifierare, en punkt, och till sist en valbar del som är medlemsvariabeln för elementet med samma id och subidentifierare som adress syntaxen. A9COLLADAAddressSyntax är en klass som innehåller de tre elementen ID, Subidentifierare samt en eventuell medlemsvariabel. Klassen har funktionalitet för att konvertera en sträng till dessa element samt tillåtelse för användaren att hämta ut nod identifikationen, subidentifieraren samt medlemsvariabeln.

A9COLLADAAddressSyntax används av inläsaren endast för att lagra information om animationskanalens mål identifikation.

4.2.7 A9AnimatedKeyFrame

A9AnimatedKeyFrame är en key-frame bestående av en tid och en transform sparas i en transformationsmatris. Klassen används utav A9AnimatedGraphicInstance samt A9AnimatedLightNode för animeringar.

4.2.8 A9AnimatedKeyFrameList

A9AnimatedKeyFrameList lagrar en lista utav A9AnimatedKeyFrames. Den lagrar även logik för vilka olika transformeringar i sina keyframes som är animerade. 6 möjliga typer är definierade i A9AnimatedKeyFrameList som alla har ett sant eller falskt värde beroende på om respektive kanal är animerad eller inte i key-frame datan. Dessa är animering av X, Y, Z, rotation kring x-axeln, rotation kring y-axeln samt rotation kring z-axeln. Klassen har funktioner för att testa om en viss animationskanal är animerad i key-framelistan. Den har även funktionalitet för att testa om alla kanaler är animerade vilket är resultatet om den inlästa animationskanalen var en transformmatris. Att spara undan denna information ger en fördel vid inläsningen av nodhierarkin för scenen eftersom informationen i de lokala transformationselementen i en nod kan adderas till de delar i en eventuell animation som inte är animerade. Ett exempel är om animationskanalen vid inläsningen endast bestod av X och Y för en viss transform enligt det omvandlade A9COLLADAAddressSyntax objektet. Det påträffade elementet i nodhierarkin som var målet för denna animation kan då lägga till sin lokala Z translation samt eventuella rotation till animationens key-frame data. A9AnimatedKeyFrameList används endast utav A9AnimatedGraphicInstance.

4.2.9 A9AnimatedGraphicInstance

A9AnimatedGraphicInstance animerar stela kroppar. A9AnimatedGraphicInstance ärver från A9GraphicInstance. A9AnimatedGraphicInstance definierar upp en A9AnimatedKeyFrameList som innehåller ett antal A9KeyFrames. Klassen A9AnimatedGraphicInstance innehåller ett A9Sequence objekt som loopar animationen från key-frame 0 till antalet key-frames i listan. Animationen interpolerar mellan två keyframes genom SLERP (Spherical LinEar inteRPolation) för rotation och LERP (LinEar Interpolation) för translation. Rotationen sker genom att först transformera de två matriserna som ska animeras till A9Quaternion objekt och låta dessa två interpolera med varandra. Då bildas ett litet A9Quaternionbarn som mest skriker och bajsar.

4.2.10 A9AnimatedLightNode

A9AnimatedLightNode används för att animera ljuskällor. Klassen ärver från A9LightNode. A9AnimatedLightNode innehåller en lista över A9GraphicInstance typer som är kopplad till transformeringen för de ljus som ljusnoden innehåller. Alla ljus inom samma nivå i renderingsträdet omvandlas i inläsaren till en A9AnimatedLightNode. Den

lista som klassen håller över A9GraphicInstance objekt för att sätta ljusets transform innebär att transformeringen kan vara animerad eftersom A9AnimatedGraphicInstance också är av typen A9GraphicInstance. Således vid inläsning av en ljusnod i COLLADA's nodhierarki kommer en ljusnod som har en transform som är animerad att bilda en A9AnimatedLightNode där den A9GraphicInstance som kopplas till ljuset är det A9AnimatedGraphicInstance objektet som skapades utav inläst animering. En ljusnod som inte är animerad bildar även den en A9AnimatedLightNode med den kopplade transformeringen som en vanlig A9GraphicInstance transform.

4.2.11 A9KeyFrame

A9KeyFrame i paketet com.agency9.agentfx.skeleton har skrivits om från dess ursprungliga struktur med en key-frame definierad som en rotationsvektor och en translationsvektor, till en ny struktur där en A9KeyFrame definieras som ett gränssnitt (interface) med en funktion som hämtar en transformationsmatris av typen A9Matrix4x4f.

4.2.12 A9EulerKeyFrame

A9EulerKeyFrame ärver från A9KeyFrame och definierar upp en key-frame med en translationsvektor och en rotationsvektor. Hämtningen av transformationen omvandlar rotationsvektorn och translationsvektorn till en matristransformation av typen A9Matrix4x4f.

4.2.13 A9MatrixKeyFrame

A9MatrixKeyFrame ärver från A9KeyFrame och innehåller en transformation sparad som en A9Matrix4x4f. En hämtning av transformationen behöver då inte omvandla informationen till en transformationsmatris eftersom transformationen internt sparas i samma typ.

4.2.14 A9Joint

A9Joint har en ändrad funktionalitet i interpolationen som direkt hämtar matris transformen via den A9KeyFrame som är hämtad från sekvens datan som anges som parameter till interpolationen.

5 Resultat

5.1 Övergripande beskrivning

5.1.1 Beskrivning av resultatet

Inläsaren stöder inläsning av COLLADA formatet med inläsning av geometri, ljus, material, texturer, animering av stela kroppar, animering av ljus samt skelettbaserad animation. En korrekt omvandling från COLLADA formatets nodhierarki till AgentFX baserade renderingsträd har uppnåtts. Inläsaren är snabb, effektiv, uppbyggd med en väldefinierad API struktur och är direkt anpassad för Agency9's behov.

5.1.2 "Piglet and Pretorean"

"Piglet and Pretorean" är en komplex sekvens skapad i Maya Unlimited 6.5. Scenen består av 2 skeletbaserade animationer, 8 stela kroppar animationer, 84 geometrier samt en animerad ljuskälla. För export till COLLADA formatet användes version 0.54 av Guillaume Laforte's och Christian Laforte's Maya COLLADA exporter [17]. Denna komplexa scen är resultatet av hårt arbete som bevisar möjligheten och styrkan i min inläsare samt formatet COLLADA.

"Piglet and Pretorean"

Fig. 9. "Piglet and Pretorean" – en komplex sekvens inläst från COLLADA formatet.

5.2 Relaterade arbeten

Open Source Java COLLADA loader [16].

6 Diskussion

6.1 Betydelse av resultaten

Betydelsen av resultaten har mycket stor betydelse för Agency9, deras grafikmotor AgentFX samt COLLADA formatet.

6.1.1 Betydelse för Agency9 och AgentFX

I och med att jag har visat att en inläsning är möjlig från COLLADA formatet till AgentFX grafikmotor via exporter från Maya kommer Agency9 att kunna erbjuda en stor del av funktionaliteten från Maya direkt kopplad till AgentFX. Genom att arbeta med COLLADA formatet i AgentFX flyttas en mycket stor del av implementationen av programmerare till grafiker och förbättrar hela ”content pipeline”. Agency9 kommer kunna erbjuda sina kunder en lösning att visa resultat genom AgentFX mycket snabbare genom COLLADA formatet och inläsarens stöd.

6.1.2 Betydelse för COLLADA formatet

COLLADA är ett relativt nytt XML baserat format. Mitt arbete visar på dess kraftfulla struktur och hur man kan skapa en implementation i en javabaserad grafikmotor. I och med att resultatet blev lyckad så ändrar det, i alla fall min egen, uppfattning om XML som format för inläsning av 3d-grafik. Genom demonstrationsexempel med AgentFX motor kommer COLLADA formatet få en större betydelse för Agency9 och deras intressenter.

6.2 Slutsats

Jag har med arbetet utav inläsaren uppnått en lösning på de problem som ställdes i mitt problemområde (se avsnitt 1.4). Lösningen är snabb och bidrar till att bygga upp en API struktur som passar ihop med AgentFX paketet. Jag har lyckats att omvandla COLLADA dokument genererade från information i en Maya scen till AgentFX renderingsträd med mycket lyckade resultat (se fig. 9) samt i en komplex sekvens byggd i Maya, exporterad till COLLADA och inläst med inläsaren för omvandling till AgentFX renderingsträd (se avsnitt 6.2.3). Jag är mycket nöjd över det arbete jag har utfört hos Agency9 och de lösningar jag har presenterat i min implementation.

6.3 Vad kunde ha gjorts bättre

Inläsaren är implementerad med hänsyn till krav av snabbhet, lättanvänd och förståelig API för vidareutveckling av COLLADA format samt att ge stöd för de vanliga funktionerna som AgentFX paketet har stöd för. Detta mål är uppnått men en hel del

förbättringar i den nuvarande inläsaren finns dock. Dessa presenteras här samt förslag till hur lösningen kunde ha gjort bättre.

6.3.1 Skeletbaserad animation

Inläsningen av skeletbaserad animation fungerar inte helt korrekt när animationen inte exporteras som bakade transformer. Detta beror på logiska luckor när en viss animationskanal inte är animerad utan endast definierad i varje joints lokala transformation. Ett problem uppstår då vissa delar utav en joint transformation är animerad medan en annan inte är det. Hur dessa animationer bygger upp varandra är inte helt fungerande i inläsaren varav skeletbaserad animation endast kommer fungera korrekt när animationerna och transformerna läses in som en matris i COLLADA dokumentet. Det är heller inte logiskt definierat hur ett skeletsystem animerad om antalet key-frames skiljer sig för några joint. En sådan skeletanimation kan inte animeras eftersom det skeletbaserade systemet i AgentFX kräver lika antal keyframes för varje joint i skeletsystemet för en korrekt animation. Den positiva sidan i skeletbaserad animation är att inläsaren klarar av joints som inte har animation i ett skeletsystem där en annan joint har animation. I dessa fall byggs key-frames baserade på varje joints lokala transformation.
Det som kunde ha gjorts bättre i skeletbaserad animation är att inrikta implementationsstödet helt och hållet på matristransformeringar för varje joint samt animationen.

En viss skeletbaserad animation kan endast finnas i en instans i scenen. Detta beror på att skinning processen inte kan dela geometri med olika animationer. Ett skelett kan bara vara kopplad till en geometri och därigenom ett material och en textur. Inläsaren stödjer dock olika skeletbaserade animationer i samma scen.

6.3.2 Stödet för ljus

Inläsarstödet för olika ljustyper och dess information är dåligt definierad. Endast punktljus (A9PointLight) kan just nu skapas i renderingsträdet i AgentFX. Detta beror på otillräcklig implementering i inläsaren. En förbättring här vore att utveckla implementationen för stöd av ”SpotLights” samt ”Directional Lights” och dessutom lägga till stöd för animation utav dessa. En ytterligare förbättring vore att inte skapa en A9AnimatedLightNode som innehåller ljus som inte är animerade utan då skapa en vanlig A9LightNode för att på så sätt inte kringgå AgentFX paketet när det inte behövs.

6.3.3 Key-frames läser in tid

COLLADA definierar upp en tid och en transform för key-frame data vilket inte är systemet som AgentFX använder. En förbättring här vore att inte alls läsa in tiden i animationsdata utan via sekvensfunktionaliteten i AgentFX skeletbaserade system bestämma iterationshastigheten istället (vilket nuvarande implementation gör).

6.3.4 Inställningar för inläsaren
Användaren av inläsaren (Reader) kan i nuvarande design och implementation inte bestämma några inställningar för hur inläsaren ska arbeta i omvandlingen. Alldeles för många beslut togs direkt i implementationen när de istället borde ha varit valbara för användaren av inläsaren. En förbättring vore att skapa en A9COLLADAConfiguration klass som tillåter funktionalitet för att bestämma hur inläsaren ska arbeta. En sådan klass skulle kunna skickas med som parameter till inläsaren och därigenom ge användaren större frihet hur renderingsträdet ska byggas upp.

6.3.5 Omvandling från Java till XML
Omvandlingen från AgentFX till ett COLLADA dokument är teoretiskt möjlig genom JAXB men implementationen stöder inte denna process och kan inte byggas ut för att göra detta. Om ett sådant behov finns för användaren av inläsaren måste hela inläsaren skrivas och designas om för detta ändamål. En sådan process låg aldrig inom problemområdet för arbetet.

6.3.6 Inläsaren dåligt testad
Inläsaren har endast testats på versioner i den enda nuvarande exportern från Maya till COLLADA [17]. Detta beror på flera orsaker. Den främsta är att arbetsstationen jag jobbade på endast hade tillgång till Maya Unlimited 6.5. En annan orsak är att den exporter som är mest utvecklad enligt COLLADA specifikationen v1.3.0 är [17]. Stödet för många AgentFX funktioner finns inte i andra exporterare. Även om detta är nämnt under begränsningar med arbetet (se avsnitt 1.1.5) så är det värt att nämna att inläsarens största begränsning är dess dåliga testning med andra format. Detta beror dock till stor del på hur pass nära specifikationen en exporter följer. Inläsaren stöder COLLADA specifikationen 1.3.0. De flesta problem med inläsaren och formatet har legat på exporter sidan som inte har följt specifikationen. Tack vare JAXB processen kommer implementationen alltid att följa specifikationen och generera felmeddelanden om dokumentet inte följer en korrekt syntax.

6.3.7 Renderingsträdet inte optimerat
Det renderingsträd som har omvandlats från COLLADA till AgentFX är inte optimerat för texturändringar. Detta gör att instanser utav samma geometri, samma material och samma textur kan genomgå en onödig texturändring eftersom de inte delar på en A9AppearanceNode. Detta problem går att lösa i inläsaren men en möjlig lösning vore att preprocessa trädstrukturen efter inläsning. Det är inte definierat om det är inläsarens uppgift att skapa ett optimalt renderingsträd eller om denna uppgift ligger i AgentFX paketet. En lösning går dock att implementera i läsaren men denna borde inte vara specifik för renderingsträdet som har omvandlats från COLLADA formatet utan av ett godtyckligt renderingsträd i AgentFX vilket tyder på att implementationen av en sådan optimering inte direkt ligger inom mitt arbetsområde.

6.4 Framtida förbättringar och utsikter

Stödet för COLLADA är på frammarsch inom många områden och för fler och fler utvecklingsverktyg. De arbeten som pågår inom COLLADA organisationen är just nu stöd för exportering av Novodex fysikmotorstruktur samt ett C++ COLLADA API för lättare inläsning av ett COLLADA dokument.

Framtida förbättringar inom mitt arbete och min inläsare vore att integrera nya versioner av COLLADA formatet till AgentFX. Agency9 har precis släppt fysikdemonstrationer baserade på en implementation utav ett nyligen utvecklad stöd för Open Dynamics Engine. En koppling med den nya fysikspecifikationen i COLLADA samt det fortsatta arbetet med ODE i AgentFX kommer att sammanfalla inom samma tidsram (sommaren 2005) samtidigt som en sådan implementation skulle sätta AgentFX motorn i mycket hög standard för framtida utveckling.

Exporterstödet, i främst Maya, uppdateras ständigt med nya versioner och funktioner som skulle kunna integreras i AgentFX. Dessa innebär bland annat multi-texturing, cg-shader stöd samt flera olika material. Några funktionaliteter som skulle kräva en enkelt implementation är exporteringar av bakat ljus med lightmaps och vertex-ljussättning samt smooth skinning som stöds i läsaren men inte är testad i AgentFX korrekt ännu.

COLLADA går även att använda till egna lösningar som är anpassade direkt för AgentFX. Detta innebär t.ex. portalsystem, script-lösningar och andra lösningar som är definierade av användaren.

6.5 Summering

I och med lanseringen av ny bättre och snabbare hårdvara inom konsol- och PC-världen krävs fler grafiker och mer ”content”-arbete än förr. Det behövs ett smidigare sätt att få information från editorer för 3d-grafik till programmerarnas verktyg, utan att ställa krav på en viss editor. COLLADA presenterar en sådan idé för att minska steget från grafikernas arbete till programmerarnas implementation. Jag har med mitt examensarbete och denna rapport visat att en inläsning från COLLADA formatet till Agency9’s grafikmotor fungerar genom 3d-editor programmet Alias Maya Unlimited 6.5 Jag har lyckats att läsa in komplexa scener och presenterat dessa scener med hjälp utav AgentFX med önskat resultat.

Referenser

[1] *"Extensible Markup Language",* W3 Consortium
URL: *http://www.w3.org/TR/REC-xml/*

[2] *"COLLADA 1.3.0 Specification",* Collada Organisation
URL: http://www.collada.org/public_forum/files/COLLADA_Specification_130.pdf

[3] *"Web Services Made Easier"* , Sun Microsystems
URL: http://java.sun.com/xml/webservices.pdf

[4] *"Core Web Programming"*, Marty Hall and Larry Brown,
Sun Microsystem/Prentice Hall, pp. 1132 - 1187
URL: http://www.corewebprogramming.com/PDF/ch23.pdf

[5] *"Java + XML = JDOM"*, Jason Hunter, Brett McLaughlin
URL: http://www.jdom.org/dist/docs/presentations/mtvjug04262000.pdf

[6] *"XML in the Java™ Platform: Overview, Roadmap and Futures"*, James Duncan Davidson
URL: http://servlet.java.sun.com/javaone/javaone2000/pdfs/TS-1502.pdf

[7] *"Issues in Generating Data Bindings for an XML Schema-Based Language"*, Eric M. Dashofy, Department of Information and Computer Science, University of California, Irvine
URL: http://www.isr.uci.edu/~edashofy/papers/xse2001.pdf

[8] *"XML – Application Programming Interfaces (APIs)"*, Jeff Hunter
URL: http://www.idevelopment.info/data/Programming/java/xml/XML_APIs.pdf

[9] *"SAX"*
URL: http://sax.sourceforge.net/

[10] *"XML on speed"*, Christian Stocker, Bitflux GmbH, PHP Conference 2004, Frankfurt
URL: http://php5.bitflux.org/xmlonspeed/title.php

[11] *"Introduction to XML"*, Dr Nikos Antonopoulos, Department of Computing, University of Surrey
URL: http://www.computing.surrey.ac.uk/personal/st/R.Peel/cs381/cs381_1_2003.pdf

[12] *"Agency9"*
URL: http://www.agency9.com/

[13] *"JAXB Reference Implementation Project"*

URL: https://jaxb.dev.java.net/

[14] *"The Java Web Services Tutorial"*
URL: http://java.sun.com/webservices/docs/1.5/tutorial/doc/index.html

[15] *"Java Code Conventions"*
URL:
http://www.idevelopment.info/data/Programming/java/coding_conventions/Java_Code_Conventions.pdf

[16] *"Open Source Java COLLADA Loader"*, Whoola, Inc.
URL: http://sourceforge.net/projects/whoola/

[17] *" COLLADA plug-in for Maya"*, Guillaume Laforte, Christian Laforte
URL: http://sourceforge.net/users/claforte/

Bilagor

Bilaga A: COLLADA Schema Specification 1.3.0

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema targetNamespace="http://www.collada.org/2005/COLLADASchema"
elementFormDefault="qualified"
  xmlns="http://www.collada.org/2005/COLLADASchema"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
  version="1.3.0">
  <!-- BEGIN COLLADA Format Schema -->
  <xs:annotation>
    <xs:documentation>
      COLLADA Format Schema
      Version 1.3.0 (May 5th, 2005)
      Copyright 2005 Sony Computer Entertainment America
      All Rights Reserved
    </xs:documentation>
  </xs:annotation>
  <!-- Root Element -->
  <xs:element name="COLLADA">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="asset" minOccurs="1" maxOccurs="1" />
        <xs:element ref="library" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="scene" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
      <xs:attribute name="version" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
  <!-- Simple Types -->
  <!-- Primitive Types -->
  <xs:simpleType name="bool">
    <xs:restriction base="xs:boolean" />
  </xs:simpleType>
  <xs:simpleType name="dateTime">
    <xs:restriction base="xs:dateTime" />
  </xs:simpleType>
  <xs:simpleType name="float">
    <xs:restriction base="xs:double" />
  </xs:simpleType>
  <xs:simpleType name="int">
    <xs:restriction base="xs:long" />
  </xs:simpleType>
  <xs:simpleType name="Name">
    <xs:restriction base="xs:Name" />
  </xs:simpleType>
  <xs:simpleType name="string">
    <xs:restriction base="xs:string" />
  </xs:simpleType>
  <xs:simpleType name="token">
    <xs:restriction base="xs:token" />
  </xs:simpleType>
  <!-- Container Types -->
```

```
<xs:simpleType name="ListOfBools">
  <!-- MCB: MS XML Designer dataview doesn't support list or union derived simpleType -->
  <xs:list itemType="xs:boolean" />
</xs:simpleType>
<xs:simpleType name="ListOfFloats">
  <!-- MCB: MS XML Designer dataview doesn't support list or union derived simpleType -->
  <xs:list itemType="xs:double" />
</xs:simpleType>
<xs:simpleType name="ListOfHexBinary">
  <!-- MCB: MS XML Designer dataview doesn't support list or union derived simpleType -->
  <xs:list itemType="xs:hexBinary" />
</xs:simpleType>
<xs:simpleType name="ListOfInts">
  <!-- MCB: MS XML Designer dataview doesn't support list or union derived simpleType -->
  <xs:list itemType="xs:long" />
</xs:simpleType>
<xs:simpleType name="ListOfNames">
  <!-- MCB: MS XML Designer dataview doesn't support list or union derived simpleType -->
  <xs:list itemType="xs:Name" />
</xs:simpleType>
<xs:simpleType name="ListOfTokens">
  <!-- MCB: MS XML Designer dataview doesn't support list or union derived simpleType -->
  <xs:list itemType="xs:token" />
</xs:simpleType>
<!-- Aggregate Types -->
<xs:simpleType name="bool2">
  <xs:restriction base="ListOfBools">
    <xs:minLength value="2" />
    <xs:maxLength value="2" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="bool3">
  <xs:restriction base="ListOfBools">
    <xs:minLength value="3" />
    <xs:maxLength value="3" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="bool4">
  <xs:restriction base="ListOfBools">
    <xs:minLength value="4" />
    <xs:maxLength value="4" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="float2">
  <xs:restriction base="ListOfFloats">
    <xs:minLength value="2" />
    <xs:maxLength value="2" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="float3">
  <xs:restriction base="ListOfFloats">
    <xs:minLength value="3" />
    <xs:maxLength value="3" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="float4">
```

```
    <xs:restriction base="ListOfFloats">
      <xs:minLength value="4" />
      <xs:maxLength value="4" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="float7">
    <xs:restriction base="ListOfFloats">
      <xs:minLength value="7" />
      <xs:maxLength value="7" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="float2x2">
    <xs:restriction base="ListOfFloats">
      <xs:minLength value="4" />
      <xs:maxLength value="4" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="float3x3">
    <xs:restriction base="ListOfFloats">
      <xs:minLength value="9" />
      <xs:maxLength value="9" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="float4x4">
    <xs:restriction base="ListOfFloats">
      <xs:minLength value="16" />
      <xs:maxLength value="16" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="int2">
    <xs:restriction base="ListOfInts">
      <xs:minLength value="2" />
      <xs:maxLength value="2" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="int3">
    <xs:restriction base="ListOfInts">
      <xs:minLength value="3" />
      <xs:maxLength value="3" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="int4">
    <xs:restriction base="ListOfInts">
      <xs:minLength value="4" />
      <xs:maxLength value="4" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="int2x2">
    <xs:restriction base="ListOfInts">
      <xs:minLength value="4" />
      <xs:maxLength value="4" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="int3x3">
    <xs:restriction base="ListOfInts">
      <xs:minLength value="9" />
```

```
      <xs:maxLength value="9" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="int4x4">
    <xs:restriction base="ListOfInts">
      <xs:minLength value="16" />
      <xs:maxLength value="16" />
    </xs:restriction>
  </xs:simpleType>
  <!-- Basic Enumerations -->
  <xs:simpleType name="ArrayTypes">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="float" />
      <xs:enumeration value="int" />
      <xs:enumeration value="Name" />
      <xs:enumeration value="token" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="FlowType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="IN" />
      <xs:enumeration value="OUT" />
      <xs:enumeration value="INOUT" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="LibraryType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="ANIMATION" />
      <xs:enumeration value="CAMERA" />
      <xs:enumeration value="CODE" />
      <xs:enumeration value="CONTROLLER" />
      <xs:enumeration value="GEOMETRY" />
      <xs:enumeration value="IMAGE" />
      <xs:enumeration value="LIGHT" />
      <xs:enumeration value="MATERIAL" />
      <xs:enumeration value="PROGRAM" />
      <xs:enumeration value="TEXTURE" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="LightType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="AMBIENT" />
      <xs:enumeration value="DIRECTIONAL" />
      <xs:enumeration value="POINT" />
      <xs:enumeration value="SPOT" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="NodeType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="JOINT" />
      <xs:enumeration value="NODE" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="UpAxisType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="X_UP" />
```

```
      <xs:enumeration value="Y_UP" />
      <xs:enumeration value="Z_UP" />
    </xs:restriction>
  </xs:simpleType>
  <!-- Complex Types -->
  <!--   none yet -->
  <!-- Dataflow Elements -->
  <xs:element name="accessor">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="param" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
      <xs:attribute name="offset" type="xs:nonNegativeInteger" default="0" />
      <xs:attribute name="source" type="xs:anyURI" use="required" />
      <xs:attribute name="stride" type="xs:nonNegativeInteger" default="1" />
    </xs:complexType>
  </xs:element>
  <xs:element name="array">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="ListOfTokens">
          <xs:attribute name="id" type="xs:ID" />
          <xs:attribute name="name" type="xs:NCName" />
          <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
          <xs:attribute name="type" type="ArrayTypes" use="required" />
          <xs:attribute name="minInclusive" type="xs:integer" default="-2147483648" />
          <xs:attribute name="maxInclusive" type="xs:integer" default="2147483647" />
          <xs:attribute name="digits" type="xs:short" default="6" />
          <xs:attribute name="magnitude" type="xs:short" default="38" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <!-- ARRAYS: typed arrays (Name, float, int, bool) -->
  <xs:element name="float_array">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="ListOfFloats">
          <xs:attribute name="id" type="xs:ID" />
          <xs:attribute name="name" type="xs:NCName" />
          <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
          <xs:attribute name="digits" type="xs:short" default="6" />
          <xs:attribute name="magnitude" type="xs:short" default="38" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="int_array">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="ListOfInts">
          <xs:attribute name="id" type="xs:ID" />
          <xs:attribute name="name" type="xs:NCName" />
          <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
```

```
            <xs:attribute name="minInclusive" type="xs:integer" default="-2147483648" />
            <xs:attribute name="maxInclusive" type="xs:integer" default="2147483647" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="Name_array">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="ListOfNames">
            <xs:attribute name="id" type="xs:ID" />
            <xs:attribute name="name" type="xs:NCName" />
            <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="bool_array">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="ListOfBools">
            <xs:attribute name="id" type="xs:ID" />
            <xs:attribute name="name" type="xs:NCName" />
            <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="input">
      <xs:complexType>
        <xs:attribute name="idx" type="xs:nonNegativeInteger" use="required" />
        <xs:attribute name="semantic" type="xs:NMTOKEN" use="required" />
        <xs:attribute name="source" type="xs:anyURI" use="required" />
      </xs:complexType>
    </xs:element>
    <xs:element name="param">
      <xs:annotation>
        <xs:documentation>Finding flow attribute unneccesary in practice. A unnamed parameter is
unbound/skipped.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="id" type="xs:ID" />
            <xs:attribute name="name" type="xs:NCName" />
            <xs:attribute name="sid" type="xs:NCName" />
            <xs:attribute name="flow" type="FlowType" />
            <xs:attribute name="semantic" type="xs:token" />
            <xs:attribute name="type" type="xs:NMTOKEN" use="required" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <!-- ARRAYS: insert a choice group with all array types in place of the old array element ref -->
    <xs:element name="source">
      <xs:complexType>
```

```
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:choice minOccurs="0" maxOccurs="1">
          <xs:element ref="array" />
          <xs:element ref="bool_array" />
          <xs:element ref="float_array" />
          <xs:element ref="int_array" />
          <xs:element ref="Name_array" />
        </xs:choice>
        <xs:element name="technique" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element ref="asset" minOccurs="0" maxOccurs="1" />
              <xs:element ref="accessor" maxOccurs="1" />
              <xs:element ref="combiner" maxOccurs="1" />
              <xs:element ref="joints" maxOccurs="1" />
              <xs:element ref="param" />
              <xs:element ref="program" maxOccurs="1" />
            </xs:choice>
            <xs:attribute name="profile" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>

  <!-- Geometry Container Elements -->
  <xs:element name="mesh">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="source" minOccurs="1" />
        <xs:element ref="vertices" minOccurs="1" maxOccurs="1" />
        <xs:element ref="lines" />
        <xs:element ref="linestrips" />
        <xs:element ref="polygons" />
        <xs:element ref="triangles" />
        <xs:element ref="trifans" />
        <xs:element ref="tristrips" />
      </xs:choice>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>
  <!-- Collation Elements -->
  <xs:element name="combiner">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="input" minOccurs="2" maxOccurs="unbounded" />
        <xs:element name="v" type="ListOfInts" minOccurs="1" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>Variable length value element. The indices form the source's output
aggregated by the number of inputs.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
```

```
      <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="joints">
    <xs:annotation>
      <xs:documentation>Joint nodes and their bind matrices.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="input" minOccurs="2" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="semantic" type="xs:NMTOKEN" use="required" />
            <xs:attribute name="source" type="xs:anyURI" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
      <xs:attribute name="count" type="xs:nonNegativeInteger" />
    </xs:complexType>
  </xs:element>
  <xs:element name="lines">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="input" />
        <xs:element ref="param" />
        <xs:element name="p" type="ListOfInts" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>Primitive element. Every two indices form a line.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
      <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
      <xs:attribute name="material" type="xs:anyURI" />
    </xs:complexType>
  </xs:element>
  <xs:element name="linestrips">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="input" />
        <xs:element ref="param" />
        <xs:element name="p" type="ListOfInts" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>Primitive element. The first two indices form a line. Each subsequent
index extends the line from the previous index.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
      <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
      <xs:attribute name="material" type="xs:anyURI" />
    </xs:complexType>
  </xs:element>

  <xs:element name="polygons">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
```

```
        <xs:element ref="input" />
        <xs:element ref="param" />
        <xs:element name="p" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>Primitive element. All the indices form a polygon.</xs:documentation>
          </xs:annotation>
          <xs:complexType mixed="true">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="h" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>Contour Separator. Primitives after this each describe a
hole.</xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
      <xs:attribute name="material" type="xs:anyURI" />
    </xs:complexType>
  </xs:element>

  <xs:element name="triangles">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="input" />
        <xs:element ref="param" />
        <xs:element name="p" type="ListOfInts" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>Primitive element. Every three indices form a
triangle.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
      <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
      <xs:attribute name="material" type="xs:anyURI" />
    </xs:complexType>
  </xs:element>
  <xs:element name="trifans">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="input" />
        <xs:element ref="param" />
        <xs:element name="p" type="ListOfInts" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>Primitive element. The 1st three indices form a triangle. Each
subsequent index forms an additional triangle reusing the first and previous
indices.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
      <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
      <xs:attribute name="material" type="xs:anyURI" />
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="tristrips">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="input" />
        <xs:element ref="param" />
        <xs:element name="p" type="ListOfInts" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>Primitive element. The 1st three indices form a triangle. Each
subsequent index forms an additional triangle reusing the previous two indices.</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:choice>
      <xs:attribute name="count" type="xs:nonNegativeInteger" use="required" />
      <xs:attribute name="material" type="xs:anyURI" />
    </xs:complexType>
  </xs:element>
  <xs:element name="vertices">
    <xs:annotation>
      <xs:documentation>Mesh or skin vertices.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="input" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="semantic" type="xs:NMTOKEN" use="required" />
            <xs:attribute name="source" type="xs:anyURI" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
      <xs:attribute name="count" type="xs:nonNegativeInteger" />
    </xs:complexType>
  </xs:element>
  <!-- Transformational Elements -->
  <xs:element name="lookat">
    <xs:annotation>
      <xs:documentation>Look-at transform (PX, PY, PZ, IX, IY, IZ, UPX, UPY,
UPZ)</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="float3x3">
          <xs:attribute name="sid" type="xs:NCName" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="matrix">
    <xs:annotation>
      <xs:documentation>Full 4x4 transformation matrix.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="float4x4">
```

```
            <xs:attribute name="sid" type="xs:NCName" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="perspective">
      <xs:annotation>
        <xs:documentation>Perspective transformation along the Z axis with the given
FOV.</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="float">
            <xs:attribute name="sid" type="xs:NCName" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="rotate">
      <xs:annotation>
        <xs:documentation>Rotate N degrees about the given axis (DX, DY, DZ,
N).</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="float4">
            <xs:attribute name="sid" type="xs:NCName" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="scale">
      <xs:annotation>
        <xs:documentation>Scale transformation (SX, SY, SZ).</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="float3">
            <xs:attribute name="sid" type="xs:NCName" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="skew">
      <xs:annotation>
        <xs:documentation>Skew N degrees between the given axes (N, DX1, DY1, DZ1, DX2, DY2,
DZ2)</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="float7">
            <xs:attribute name="sid" type="xs:NCName" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
```

```
  <xs:element name="translate">
    <xs:annotation>
      <xs:documentation>Translate transformation (DX, DY, DZ).</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="float3">
          <xs:attribute name="sid" type="xs:NCName" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <!-- Material and Shading Elements -->
  <xs:element name="image">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="ListOfHexBinary">
          <xs:attribute name="id" type="xs:ID" />
          <xs:attribute name="name" type="xs:NCName" />
          <xs:attribute name="source" type="xs:anyURI" />
          <xs:attribute name="format" type="xs:string" />
          <xs:attribute name="height" type="xs:nonNegativeInteger" />
          <xs:attribute name="width" type="xs:nonNegativeInteger" />
          <xs:attribute name="depth" type="xs:nonNegativeInteger" default="1" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="light">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="asset" minOccurs="0" maxOccurs="1" />
        <xs:element ref="param" minOccurs="1" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
      <xs:attribute name="type" type="LightType" default="POINT" />
    </xs:complexType>
  </xs:element>
  <xs:element name="material">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="asset" minOccurs="0" maxOccurs="1" />
        <xs:element ref="param" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="shader" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>
  <xs:element name="pass">
    <xs:complexType>
      <xs:choice>
        <xs:sequence>
          <xs:element ref="param" minOccurs="0" maxOccurs="unbounded" />
          <xs:element name="input" minOccurs="0" maxOccurs="unbounded" >
```

```
            <xs:complexType>
              <xs:attribute name="semantic" type="xs:NMTOKEN" use="required" />
              <xs:attribute name="source" type="xs:anyURI" use="required" />
            </xs:complexType>
          </xs:element>
          <xs:element ref="program" minOccurs="0" />
        </xs:sequence>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="shader">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="param" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="technique" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element ref="asset" minOccurs="0" maxOccurs="1" />
              <xs:element ref="param" />
              <xs:element ref="pass" />
              <xs:element ref="program" maxOccurs="1" />
            </xs:choice>
            <xs:attribute name="profile" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>
  <xs:element name="texture">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="asset" minOccurs="0" maxOccurs="1" />
        <xs:element ref="param" minOccurs="0" maxOccurs="unbounded" />
        <xs:element name="technique" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element ref="asset" minOccurs="0" maxOccurs="1" />
              <xs:element name="input" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:attribute name="semantic" type="xs:NMTOKEN" use="required" />
                  <xs:attribute name="source" type="xs:anyURI" use="required" />
                </xs:complexType>
              </xs:element>
              <xs:element ref="param" />
              <xs:element ref="program" maxOccurs="1" />
            </xs:choice>
            <xs:attribute name="profile" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>
```

```
<!-- Procedural Elements -->
<xs:element name="code">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="id" type="xs:ID" />
        <xs:attribute name="lang" type="xs:NMTOKEN" use="required" />
        <xs:attribute name="profile" type="xs:string" />
        <xs:attribute name="semantic" type="xs:NMTOKEN" />
        <xs:attribute name="url" type="xs:anyURI" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="entry">
  <xs:complexType>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="param" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="id" type="xs:ID" />
              <xs:attribute name="name" type="xs:NCName" use="required" />
              <xs:attribute name="qualifier" type="xs:NMTOKEN" use="required" />
              <xs:attribute name="semantic" type="xs:token" use="required" />
              <xs:attribute name="type" type="xs:NMTOKEN" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="function" type="xs:NMTOKEN" use="required" />
    <xs:attribute name="semantic" type="xs:NMTOKEN" />
  </xs:complexType>
</xs:element>
<xs:element name="program">
  <xs:complexType>
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element ref="asset" minOccurs="0" maxOccurs="1" />
      <xs:element ref="param" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="entry" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="code" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" />
    <xs:attribute name="name" type="xs:Name" />
    <xs:attribute name="url" type="xs:anyURI" />
  </xs:complexType>
</xs:element>
<!-- Object Elements -->
<xs:element name="camera">
  <xs:complexType>
    <xs:annotation>
      <xs:documentation>A camera's output is defined its imager.</xs:documentation>
    </xs:annotation>
    <xs:choice minOccurs="1" maxOccurs="unbounded">
```

```
        <xs:element name="technique">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="asset" minOccurs="0" maxOccurs="1" />
              <xs:element name="optics">
                <xs:complexType>
                  <xs:choice>
                    <xs:element ref="program" />
                  </xs:choice>
                </xs:complexType>
              </xs:element>
              <xs:element name="imager" minOccurs="0">
                <xs:complexType>
                  <xs:choice>
                    <xs:element ref="program" />
                  </xs:choice>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="profile" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>
  <xs:element name="instance">
    <xs:complexType>
      <xs:attribute name="url" type="xs:anyURI" use="required" />
    </xs:complexType>
  </xs:element>
  <!-- Animation Elements -->
  <xs:element name="channel">
    <xs:complexType>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
      <xs:attribute name="source" type="xs:anyURI" use="required" />
      <xs:attribute name="target" type="xs:token" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="sampler">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="input" minOccurs="1" maxOccurs="unbounded" >
          <xs:complexType>
            <xs:attribute name="semantic" type="xs:NMTOKEN" use="required" />
            <xs:attribute name="source" type="xs:anyURI" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>
  <!-- Controller Elements -->
```

```
  <xs:element name="skin">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="source" />
        <xs:element ref="vertices" minOccurs="1" maxOccurs="1" />
      </xs:choice>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>
  <!-- Meta Elements -->
  <xs:element name="asset">
    <xs:annotation>
      <xs:documentation>
        Asset management information.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="author" type="xs:string" />
        <xs:element name="authoring_tool" type="xs:string" maxOccurs="1" />
        <xs:element name="created" type="xs:dateTime" maxOccurs="1" />
        <xs:element name="modified" type="xs:dateTime" minOccurs="1" maxOccurs="1" />
        <xs:element name="revision" type="xs:string" maxOccurs="1" />
        <xs:element name="source_data" type="xs:anyURI" maxOccurs="1" />
        <xs:element name="copyright" type="xs:string" />
        <xs:element name="title" type="xs:string" maxOccurs="1" />
        <xs:element name="subject" type="xs:string" maxOccurs="1" />
        <xs:element name="keywords" type="xs:string" />
        <xs:element name="comments" type="xs:string" />
        <xs:element name="up_axis" type="UpAxisType" maxOccurs="1" default="Y_UP" />
        <xs:element name="unit" maxOccurs="1">
          <xs:complexType>
            <xs:attribute name="meter" type="xs:double" default="1.000" />
            <xs:attribute name="name" type="xs:NMTOKEN" default="meter" />
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="extra">
    <xs:annotation>
      <xs:documentation>
        A bag of techniques.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType >
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="technique" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element ref="asset" minOccurs="0" maxOccurs="1" />
              <xs:element ref="param" />
            </xs:choice>
            <xs:attribute name="profile" type="xs:string" use="required" />
          </xs:complexType>
```

```
        </xs:element>
      </xs:choice>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
      <xs:attribute name="type" type="xs:NMTOKEN" />
    </xs:complexType>
  </xs:element>
  <!-- Hierarchical Elements -->
  <xs:element name="boundingbox">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="min" type="float3" minOccurs="1" maxOccurs="1" />
        <xs:element name="max" type="float3" minOccurs="1" maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="node">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="lookat" />
        <xs:element ref="matrix" />
        <xs:element ref="perspective" />
        <xs:element ref="rotate" />
        <xs:element ref="scale" />
        <xs:element ref="skew" />
        <xs:element ref="translate" />
        <xs:element ref="boundingbox" maxOccurs="1" />
        <xs:element ref="instance" />
        <xs:element ref="node" />
        <xs:element ref="extra" />
      </xs:choice>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
      <xs:attribute name="type" type="NodeType" default="NODE" />
    </xs:complexType>
  </xs:element>
  <xs:element name="scene">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="lookat" />
        <xs:element ref="matrix" />
        <xs:element ref="perspective" />
        <xs:element ref="rotate" />
        <xs:element ref="scale" />
        <xs:element ref="skew" />
        <xs:element ref="translate" />
        <xs:element ref="boundingbox" maxOccurs="1" />
        <xs:element ref="node" />
        <xs:element ref="extra" />
      </xs:choice>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>
  <!-- Categorical Elements -->
  <xs:element name="animation">
```

```
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element ref="source" maxOccurs="unbounded" />
        <xs:element ref="sampler" maxOccurs="unbounded" />
        <xs:element ref="channel" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>

  <xs:element name="controller">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="skin" minOccurs="1" maxOccurs="1" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
      <xs:attribute name="target" type="xs:IDREF" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="geometry">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="mesh" minOccurs="1" maxOccurs="1" />
        <xs:element ref="extra" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
    </xs:complexType>
  </xs:element>
  <xs:element name="library">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="animation" />
        <xs:element ref="camera" />
        <xs:element ref="code" />
        <xs:element ref="controller" />
        <xs:element ref="geometry" />
        <xs:element ref="image" />
        <xs:element ref="light" />
        <xs:element ref="material" />
        <xs:element ref="program" />
        <xs:element ref="texture" />
      </xs:choice>
      <xs:attribute name="id" type="xs:ID" />
      <xs:attribute name="name" type="xs:NCName" />
      <xs:attribute name="type" type="LibraryType" use="required" />
    </xs:complexType>
  </xs:element>
  <!-- END Format Schema -->
</xs:schema>
```

Bilaga B: XJC Settings file (”collada.xjb”)

```
<jxb:bindings version="1.0"
    xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <jxb:bindings schemaLocation="http://www.collada.org/2005/COLLADASchema.xsd"
node="/xs:schema">

<!-- ******************** Global binding settings ******************** -->
    <jxb:globalBindings
        fixedAttributeAsConstantProperty="false"
        collectionType="java.util.ArrayList"
        typesafeEnumBase="xs:NMTOKEN"
        choiceContentProperty="true"
        typesafeEnumMemberName="generateError"
        bindingStyle="elementBinding"
        enableFailFastCheck="false"
        generateIsSetMethod="false"
        underscoreBinding="asWordSeparator">

      <!-- Global java type bindings -->
      <jxb:javaType name="int" xmlType="xs:nonNegativeInteger"
        parseMethod="javax.xml.bind.DatatypeConverter.parseInt"
        printMethod="javax.xml.bind.DatatypeConverter.printInt"/>

    </jxb:globalBindings>

<!-- ******************** Schema binding settings ******************** -->
    <jxb:schemaBindings>
      <jxb:package name="com.agency9.agentfx.io.collada">
        <jxb:javadoc><![CDATA[<body>Package level
            documentation for generated package com.agency9.agentfx.io.collada</body>]]>
        </jxb:javadoc>
      </jxb:package>
      <jxb:nameXmlTransform>
        <jxb:elementName prefix="A9COLLADA" suffix="Element"/>
        <jxb:typeName prefix="A9COLLADA" suffix="Constants"/>
        <jxb:anonymousTypeName prefix="A9COLLADA" suffix="Type"/>
      </jxb:nameXmlTransform>
    </jxb:schemaBindings>



<!-- ******************** Primitive Types goes here ******************** -->

<!-- ******************** Container Types goes here ******************** -->
    <jxb:bindings node="//xs:simpleType[@name='ListOfFloats']">
      <jxb:javaType name="com.agency9.buffer.A9FloatBuffer"
          parseMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseFloatBuffer"
          printMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseString"/>
    </jxb:bindings>
```

```
<!-- ******************** Aggregate Types goes here ******************** -->
    <jxb:bindings node="//xs:simpleType[@name='bool2']">
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='bool3']">
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='bool4']">
    </jxb:bindings>


    <jxb:bindings node="//xs:simpleType[@name='float2']">
      <jxb:javaType name="com.agency9.math.vectormath.A9Vector2f"
          parseMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseFloat2"
          printMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseString"/>
    </jxb:bindings>


    <jxb:bindings node="//xs:simpleType[@name='float3']">
      <jxb:javaType name="com.agency9.math.vectormath.A9Vector3f"
          parseMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseFloat3"
          printMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseString"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='float4']">
      <jxb:javaType name="com.agency9.math.vectormath.A9Vector4f"
          parseMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseFloat4"
          printMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseString"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='float7']">
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='float2x2']">
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='float3x3']">
      <jxb:javaType name="com.agency9.math.vectormath.A9Matrix3x3f"
          parseMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseFloat3x3"
          printMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseString"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='float4x4']">
      <jxb:javaType name="com.agency9.math.vectormath.A9Matrix4x4f"
          parseMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseFloat4x4"
          printMethod="com.agency9.agentfx.io.collada.A9COLLADAParser.parseString"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='int2']">
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='int3']">
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='int4']">
    </jxb:bindings>
```

```
    <jxb:bindings node="//xs:simpleType[@name='int2x2']">
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='int3x3']">
    </jxb:bindings>

    <jxb:bindings node="//xs:simpleType[@name='int4x4']">
    </jxb:bindings>

<!-- ******************** Basic Enumerations ******************** -->
    <jxb:bindings node="//xs:simpleType[@name='ArrayTypes']">
      <jxb:typesafeEnumClass name="A9COLLADAArrayConstants">
      </jxb:typesafeEnumClass>
    </jxb:bindings>
    <jxb:bindings node="//xs:simpleType[@name='FlowType']">
      <jxb:typesafeEnumClass name="A9COLLADAFlowConstants">
      </jxb:typesafeEnumClass>
    </jxb:bindings>
    <jxb:bindings node="//xs:simpleType[@name='LibraryType']">
      <jxb:typesafeEnumClass name="A9COLLADALibraryConstants">
      </jxb:typesafeEnumClass>
    </jxb:bindings>
    <jxb:bindings node="//xs:simpleType[@name='LightType']">
      <jxb:typesafeEnumClass name="A9COLLADALightConstants">
      </jxb:typesafeEnumClass>
    </jxb:bindings>
    <jxb:bindings node="//xs:simpleType[@name='NodeType']">
      <jxb:typesafeEnumClass name="A9COLLADANodeConstants">
      </jxb:typesafeEnumClass>
    </jxb:bindings>
    <jxb:bindings node="//xs:simpleType[@name='UpAxisType']">
      <jxb:typesafeEnumClass name="A9COLLADAUpAxisConstants">
      </jxb:typesafeEnumClass>
    </jxb:bindings>




<!-- ******************** Dataflow Elements ******************** -->
    <jxb:bindings node="//xs:element[@name='COLLADA']">
      <jxb:class name="A9COLLADACOLLADAElement"/>
      <jxb:bindings node=".//xs:sequence">
        <jxb:bindings node="//xs:element[@ref='library']">
          <jxb:property name="Libraries"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='accessor']">
      <jxb:class name="A9COLLADAAccessorElement"/>
      <jxb:bindings node=".//xs:complexType">
        <jxb:bindings node=".//xs:sequence">
          <jxb:property name="Params"/>
        </jxb:bindings>
      </jxb:bindings>
```

```
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='array']">
      <jxb:class name="A9COLLADAArrayElement"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='float_array']">
      <jxb:class name="A9COLLADAFloatArrayElement"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='int_array']">
      <jxb:class name="A9COLLADAIntArrayElement"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='Name_array']">
      <jxb:class name="A9COLLADANameArrayElement"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='bool_array']">
      <jxb:class name="A9COLLADABoolArrayElement"/>
    </jxb:bindings>

    <jxb:bindings node="xs:element[@name='input'][1]">
      <jxb:class name="A9COLLADAInputElement"/>
    </jxb:bindings>

    <jxb:bindings node="xs:element[@name='param'][1]">
      <jxb:class name="A9COLLADAParamElement"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='source']">
      <jxb:class name="A9COLLADASourceElement"/>
      <jxb:bindings node="xs:complexType[1]">
        <jxb:bindings node="xs:choice[1]">
          <jxb:property name="SourceElements"/>
          <jxb:bindings node=".//xs:element[@name='technique']">
            <jxb:class name="A9SourceTechnique"/>
            <jxb:bindings node=".//xs:choice">
              <jxb:property name="TechniqueElements"/>
            </jxb:bindings>
          </jxb:bindings>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>

<!-- ******************** Geometry Container Elements ******************** -->
    <jxb:bindings node="//xs:element[@name='mesh']">
      <jxb:class name="A9COLLADAMeshElement"/>
      <jxb:bindings node=".//xs:choice">
        <jxb:property name="MeshElements"/>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='combiner']">
      <jxb:class name="A9COLLADACombinerElement"/>
      <jxb:bindings node=".//xs:complexType">
```

```
      <jxb:bindings node=".//xs:sequence">
        <jxb:bindings node=".//xs:element[@ref='input']">
          <jxb:property name="InputElements"/>
        </jxb:bindings>
        <jxb:bindings node=".//xs:element[@name='v']">
          <jxb:class name="A9VIndicies"/>
          <jxb:property name="Indicies"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='joints']">
    <jxb:class name="A9COLLADAJointsElement"/>
    <jxb:bindings node=".//xs:sequence">
        <jxb:property name="JointsInputs"/>
        <jxb:bindings node=".//xs:element[@name='input']">
          <jxb:class name="A9JointsInput"/>
        </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='lines']">
    <jxb:class name="A9COLLADALinesElement"/>
    <jxb:bindings node=".//xs:choice">
        <jxb:property name="LinesElements"/>
        <jxb:bindings node=".//xs:element[@name='p']">
          <jxb:class name="A9LinesPrimitive"/>
        </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='linestrips']">
    <jxb:class name="A9COLLADALineStripsElement"/>
    <jxb:bindings node=".//xs:choice">
      <jxb:property name="LineStripsElements"/>
      <jxb:bindings node=".//xs:element[@name='p']">
        <jxb:class name="A9LineStripsPrimitive"/>
      </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='polygons']">
    <jxb:class name="A9COLLADAPolygonsElement"/>
    <jxb:bindings node="xs:complexType[1]">
      <jxb:bindings node="xs:choice[1]">
        <jxb:property name="PolygonsElements"/>
        <jxb:bindings node=".//xs:element[@name='p']">
          <jxb:class name="A9PolygonsPrimitive"/>

          <jxb:bindings node=".//xs:choice">
            <jxb:bindings node=".//xs:element[@name='h']">
              <jxb:class name="A9PolygonHole"/>
            </jxb:bindings>
          </jxb:bindings>
```

```
          </jxb:bindings>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='triangles']">
      <jxb:class name="A9COLLADATrianglesElement"/>
      <jxb:bindings node=".//xs:choice">
        <jxb:property name="TrianglesElements"/>
        <jxb:bindings node=".//xs:element[@name='p']">
          <jxb:class name="A9TrianglesPrimitive"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='trifans']">
      <jxb:class name="A9COLLADATriFansElement"/>
      <jxb:bindings node=".//xs:choice">
        <jxb:property name="TriFansElements"/>
        <jxb:bindings node=".//xs:element[@name='p']">
          <jxb:class name="A9TriFansPrimitive"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='tristrips']">
      <jxb:class name="A9COLLADATriStripsElement"/>
      <jxb:bindings node=".//xs:choice">
        <jxb:property name="TriStripsElements"/>
        <jxb:bindings node=".//xs:element[@name='p']">
          <jxb:class name="A9TriStripsPrimitive"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='vertices']">
      <jxb:class name="A9COLLADAVerticesElement"/>
      <jxb:bindings node=".//xs:sequence">
        <jxb:property name="VerticesInputs"/>
        <jxb:bindings node=".//xs:element[@name='input']">
          <jxb:class name="A9VerticesInput"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>

<!-- ******************** Transformational Elements ******************** -->
    <jxb:bindings node="//xs:element[@name='lookat']">
      <jxb:class name="A9COLLADALookAtElement"/>
      <jxb:bindings node=".//xs:attribute[@name='sid']">
        <jxb:property name="SubIdentifier"/>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='matrix']">
      <jxb:class name="A9COLLADAMatrixElement"/>
      <jxb:bindings node=".//xs:attribute[@name='sid']">
```

```
        <jxb:property name="SubIdentifier"/>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='perspective']">
      <jxb:class name="A9COLLADAPerspectiveElement"/>
      <jxb:bindings node=".//xs:attribute[@name='sid']">
        <jxb:property name="SubIdentifier"/>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='rotate']">
      <jxb:class name="A9COLLADARotateElement"/>
      <jxb:bindings node=".//xs:attribute[@name='sid']">
        <jxb:property name="SubIdentifier"/>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='scale']">
      <jxb:class name="A9COLLADAScaleElement"/>
      <jxb:bindings node=".//xs:attribute[@name='sid']">
        <jxb:property name="SubIdentifier"/>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='skew']">
      <jxb:class name="A9COLLADASkewElement"/>
      <jxb:bindings node=".//xs:attribute[@name='sid']">
        <jxb:property name="SubIdentifier"/>
      </jxb:bindings>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='translate']">
      <jxb:class name="A9COLLADATranslateElement"/>
      <jxb:bindings node=".//xs:attribute[@name='sid']">
        <jxb:property name="SubIdentifier"/>
      </jxb:bindings>
    </jxb:bindings>

<!-- ******************** Material and Shading Elements ******************** -->
    <jxb:bindings node="//xs:element[@name='image']">
      <jxb:class name="A9COLLADAImageElement"/>
    </jxb:bindings>

    <jxb:bindings node="//xs:element[@name='light']">
      <jxb:class name="A9COLLADALightElement"/>
      <jxb:bindings node=".//xs:complexType">
        <jxb:bindings node=".//xs:sequence">
          <jxb:bindings node=".//xs:element[@ref='asset']">
            <jxb:property name="Asset"/>
          </jxb:bindings>
          <jxb:bindings node=".//xs:element[@ref='param']">
            <jxb:property name="Params"/>
          </jxb:bindings>
        </jxb:bindings>
      </jxb:bindings>
```

```
</jxb:bindings>

<jxb:bindings node="//xs:element[@name='material']">
  <jxb:class name="A9COLLADAMaterialElement"/>
  <jxb:bindings node=".//xs:sequence">
    <jxb:bindings node=".//xs:element[@ref='asset']">
      <jxb:property name="Asset"/>
    </jxb:bindings>
    <jxb:bindings node=".//xs:element[@ref='param']">
      <jxb:property name="Params"/>
    </jxb:bindings>
    <jxb:bindings node=".//xs:element[@ref='shader']">
      <jxb:property name="Shaders"/>
    </jxb:bindings>
  </jxb:bindings>
</jxb:bindings>

<jxb:bindings node="//xs:element[@name='pass']">
  <jxb:class name="A9COLLADAPassElement"/>
  <jxb:bindings node=".//xs:sequence">

    <jxb:bindings node=".//xs:element[@ref='param']">
      <jxb:property name="Params"/>
    </jxb:bindings>

    <jxb:bindings node=".//xs:element[@name='input']">
      <jxb:class name="A9PassInputElement"/>
      <jxb:property name="PassInputs"/>
    </jxb:bindings>

    <jxb:bindings node=".//xs:element[@ref='program']">
      <jxb:property name="Programs"/>
    </jxb:bindings>

  </jxb:bindings>
</jxb:bindings>

<jxb:bindings node="//xs:element[@name='shader']">
  <jxb:class name="A9COLLADAShaderElement"/>
  <jxb:bindings node=".//xs:sequence">

    <jxb:bindings node="xs:element[1][@ref='param']">
      <jxb:property name="Params"/>
    </jxb:bindings>

    <jxb:bindings node=".//xs:element[@name='technique']">
      <jxb:class name="A9ShaderTechniqueElement"/>

      <jxb:bindings node=".//xs:choice">
        <jxb:property name="TechniqueElements"/>
      </jxb:bindings>

    </jxb:bindings>

  </jxb:bindings>
</jxb:bindings>
```

```
<jxb:bindings node="//xs:element[@name='texture']">
  <jxb:class name="A9COLLADATextureElement"/>
  <jxb:bindings node=".//xs:sequence">

    <jxb:bindings node="xs:element[1][@ref='asset']">
      <jxb:property name="Asset"/>
    </jxb:bindings>

    <jxb:bindings node="xs:element[@ref='param']">
      <jxb:property name="Params"/>
    </jxb:bindings>

    <jxb:bindings node=".//xs:element[@name='technique']">
      <jxb:class name="A9TextureTechniqueElement"/>
      <jxb:property name="Techniques"/>
      <jxb:bindings node=".//xs:choice">
        <jxb:property name="TechniqueElements"/>
        <jxb:bindings node=".//xs:element[@name='input']">
          <jxb:class name="Input"/>
          <jxb:property name="TextureInput"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>

  </jxb:bindings>

</jxb:bindings>

<jxb:bindings node="//xs:element[@name='code']">
  <jxb:class name="A9COLLADACodeElement"/>
  <jxb:bindings node=".//xs:attribute[@name='lang']">
    <jxb:property name="Language"/>
  </jxb:bindings>
</jxb:bindings>

<jxb:bindings node="//xs:element[@name='entry']">
  <jxb:class name="A9COLLADAEntryElement"/>
  <jxb:bindings node=".//xs:element[@name='param']">
    <jxb:property name="ParamElements"/>
  </jxb:bindings>
</jxb:bindings>

<jxb:bindings node="//xs:element[@name='program']">
  <jxb:class name="A9COLLADAProgramElement"/>
  <jxb:bindings node=".//xs:complexType">
    <jxb:bindings node=".//xs:sequence">
      <jxb:property name="ProgramElements"/>
    </jxb:bindings>
  </jxb:bindings>

</jxb:bindings>

<jxb:bindings node="//xs:element[@name='camera']">
  <jxb:class name="A9COLLADACameraElement"/>
  <jxb:bindings node=".//xs:element[@name='technique']">
```

```
      <jxb:property name="Techniques"/>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='instance']">
    <jxb:class name="A9COLLADAInstanceElement"/>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='channel']">
    <jxb:class name="A9COLLADAChannelElement"/>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='sampler']">
    <jxb:class name="A9COLLADASamplerElement"/>
    <jxb:bindings node=".//xs:element[@name='input']">
      <jxb:class name="A9SamplerInput"/>
      <jxb:property name="Inputs"/>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='skin']">
    <jxb:class name="A9COLLADASkinElement"/>
    <jxb:bindings node=".//xs:choice">
      <jxb:property name="SkinElements"/>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='asset']">
    <jxb:class name="A9COLLADAAssetElement"/>
    <jxb:bindings node="//xs:element[@name='author']">
      <jxb:class name="A9AssetAuthorInfo"/>
      <jxb:property name="Author"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='authoring_tool']">
      <jxb:class name="A9AssetAuthoringToolInfo"/>
      <jxb:property name="Tool"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='created']">
      <jxb:class name="A9AssetCreatedInfo"/>
      <jxb:property name="Date"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='modified']">
      <jxb:class name="A9AssetModifiedInfo"/>
      <jxb:property name="Date"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='revision']">
      <jxb:class name="A9AssetRevisionInfo"/>
      <jxb:property name="Revision"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='source_data']">
      <jxb:class name="A9AssetSourceDataInfo"/>
      <jxb:property name="URI"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='copyright']">
      <jxb:class name="A9AssetCopyrightInfo"/>
      <jxb:property name="Copyright"/>
```

```
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='title']">
      <jxb:class name="A9AssetTitleInfo"/>
      <jxb:property name="Title"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='subject']">
      <jxb:class name="A9AssetSubjectInfo"/>
      <jxb:property name="Subject"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='keywords']">
      <jxb:class name="A9AssetKeywordInfo"/>
      <jxb:property name="Keyword"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='comments']">
      <jxb:class name="A9AssetCommentInfo"/>
      <jxb:property name="Comment"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='up_axis']">
      <jxb:class name="A9AssetUpAxisInfo"/>
      <jxb:property name="Type"/>
    </jxb:bindings>
    <jxb:bindings node="//xs:element[@name='unit']">
      <jxb:class name="A9AssetUnitInfo"/>
      <jxb:property name="Unit"/>
    </jxb:bindings>
  </jxb:bindings>


  <jxb:bindings node="//xs:element[@name='extra']">
    <jxb:class name="A9COLLADAExtraElement"/>
    <jxb:bindings node=".//xs:element[@name='technique']">
      <jxb:property name="A9ExtraTechniques"/>
      <jxb:bindings node=".//xs:complexType">
        <jxb:bindings node=".//xs:choice">
          <jxb:property name="TechniqueElements"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='boundingbox']">
    <jxb:class name="A9COLLADABoundingBoxElement"/>
    <jxb:bindings node=".//xs:complexType">
      <jxb:bindings node=".//xs:sequence">
        <jxb:bindings node=".//xs:element[@name='min']">
          <jxb:property name="A9BoundingBoxMin"/>
        </jxb:bindings>
        <jxb:bindings node=".//xs:element[@name='max']">
          <jxb:property name="A9BoundingBoxMax"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='node']">
    <jxb:class name="A9COLLADANodeElement"/>
```

```
    <jxb:bindings node=".//xs:complexType">
      <jxb:bindings node=".//xs:choice">
        <jxb:property name="NodeElements"/>
      </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='scene']">
    <jxb:class name="A9COLLADASceneElement"/>
    <jxb:bindings node=".//xs:complexType">
      <jxb:bindings node=".//xs:choice">
        <jxb:property name="SceneElements"/>
      </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='animation']">
    <jxb:class name="A9COLLADAAnimationElement"/>
    <jxb:bindings node=".//xs:complexType">
      <jxb:bindings node=".//xs:sequence">
        <jxb:property name="AnimationElements"/>
      </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='controller']">
    <jxb:class name="A9COLLADAControllerElement"/>
    <jxb:bindings node=".//xs:complexType">
      <jxb:bindings node=".//xs:sequence">
        <jxb:bindings node="//xs:element[@ref='skin']">
          <jxb:property name="Skin"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='geometry']">
    <jxb:class name="A9COLLADAGeometryElement"/>
    <jxb:bindings node=".//xs:complexType">
      <jxb:bindings node=".//xs:sequence">
        <jxb:bindings node=".//xs:element[@ref='mesh']">
          <jxb:property name="Mesh"/>
        </jxb:bindings>
        <jxb:bindings node=".//xs:element[@ref='extra']">
          <jxb:property name="Extra"/>
        </jxb:bindings>
      </jxb:bindings>
    </jxb:bindings>
  </jxb:bindings>

  <jxb:bindings node="//xs:element[@name='library']">
    <jxb:class name="A9COLLADALibraryElement"/>
    <jxb:bindings node=".//xs:complexType">
      <jxb:bindings node=".//xs:choice">
        <jxb:property name="LibraryElements"/>
      </jxb:bindings>
```

```
        </jxb:bindings>
      </jxb:bindings>

    </jxb:bindings>
</jxb:bindings>
```

Bilaga C: Cube.xml

```
<?xml version="1.0" encoding="utf-8"?>
<COLLADA xmlns="http://www.collada.org/2005/COLLADASchema" version="1.2.0">
 <asset>
  <revision>1.0</revision>
  <authoring_tool>EQUINOX-3D  Collada exporter v0.7.4</authoring_tool>
  <modified>2004-11-30T22:10:18Z</modified>
 </asset>
 <library type="LIGHT">
  <light id="Lt-Light" name="light">
   <param name="COLOR" type="float3" flow="IN">1 1 1</param>
  </light>
 </library>
 <library type="CAMERA">
  <camera id="camera" name="PerspCamera">
   <technique profile="COMMON">
    <optics>
     <program url="PERSPECTIVE">
      <param name="YFOV" type="float" flow="IN">36.000000</param>
      <param name="ZNEAR" type="float" flow="IN">0.100000</param>
      <param name="ZFAR" type="float" flow="IN">32767.0</param>
     </program>
    </optics>
   </technique>
  </camera>
 </library>
 <library type="MATERIAL">
  <material name="Blue" id="Blue">
   <shader>
    <technique profile="COMMON">
     <pass>
      <program url="PHONG">
       <param name="AMBIENT" type="float3" flow="IN">0.000000 0.000000
0.000000</param>
       <param name="DIFFUSE" type="float3" flow="IN">0.137255 0.403922 0.870588</param>
       <param name="SPECULAR" type="float3" flow="IN">0.500000 0.500000
0.500000</param>
       <param name="SHININESS" type="float" flow="IN">16.000000</param>
      </program>
     </pass>
    </technique>
   </shader>
  </material>
 </library>
 <library type="GEOMETRY">
  <geometry id="box" name="box">
   <mesh>
    <source id="box-Pos">
     <array id="box-Pos-array" type="float" count="24">
-0.5  0.5  0.5
 0.5  0.5  0.5
-0.5 -0.5  0.5
 0.5 -0.5  0.5
```

```
-0.5  0.5 -0.5
 0.5  0.5 -0.5
-0.5 -0.5 -0.5
 0.5 -0.5 -0.5
</array>
        <technique profile="COMMON">
          <accessor source="#box-Pos-array" count="8" stride="3">
            <param name="X" type="float" flow="OUT"/>
            <param name="Y" type="float" flow="OUT"/>
            <param name="Z" type="float" flow="OUT"/>
          </accessor>
        </technique>
      </source>
      <vertices id="box-Vtx">
        <input semantic="POSITION" source="#box-Pos"/>
      </vertices>
      <polygons count="6" material="#Blue">
        <input semantic="VERTEX" source="#box-Vtx" idx="0"/>
        <p>0 2 3 1</p>
        <p>0 1 5 4</p>
        <p>6 7 3 2</p>
        <p>0 4 6 2</p>
        <p>3 7 5 1</p>
        <p>5 7 6 4</p>
      </polygons>
    </mesh>
  </geometry>
 </library>
 <scene id="DefaultScene">
   <node id="Camera">
     <instance url="#camera"/>
     <lookat>
-4.277485 3.338552 6.550171
0.011243 0.125787 -0.053892
0.205746 0.925901 -0.316822
</lookat>
   </node>
   <node id="Light" name="Light">
     <instance url="#Lt-Light"/>
     <translate>-5.000000 10.000000 4.000000</translate>
     <rotate>0 0 1 0</rotate>
     <rotate>0 1 0 0</rotate>
     <rotate>1 0 0 0</rotate>
   </node>
   <node id="Box" name="Box">
     <instance url="#box"/>
     <rotate>0 0 1 0</rotate>
     <rotate>0 1 0 0</rotate>
     <rotate>1 0 0 0</rotate>
   </node>
 </scene>
</COLLADA>
```