An introduction to X3D

X3D is a royalty-free open standard's file format and run-time architecture to represent and communicate 3D scenes and objects using **XML**. It is an ISO ratified standard that provides a system for the storage, retrieval and playback of real time graphics content embedded in applications, all within an open architecture to support a wide array of domains and user scenarios. X3D is also intended to be a universal interchange format for integrated 3D graphics and multimedia. The data integration capacities and the set of componentized features of X3D is expanding allowing a 3D immersive virtual experience for the Web and mobile devices across Geospacial, Augmented Reality, CAD, medical and other markets.

The development of real-time communication of 3D data across all applications and network applications has evolved from its beginnings as the Virtual Reality Modeling Language (**VRML**) to the considerably more mature and refined X3D standard.
Although VRML has been superseded by the X3D standard, VRML files can still be read by any X3D-savvy application.

- VRML text files use a **.wrl** extension
- X3D text files written in XML use a **.x3d** extension (but the .x3dv and .x3db extensions are also possible for the files using the Classic VRML encoding and the Binary encoding, respectivelly)

X3D/VRML are supported by the Web3D Consortium (see http://www.web3d.org/)

We can construct X3D/VRML files using:

- **A text editor**

Pros:
- No new software to buy
- Access to all X3D/VRML features
- Detailed control of world efficiency

Cons:
- Hard to author complex 3D shapes
- Requires knowledge of X3D/VRML syntax

- **A world builder application**

Pros:
- Easy 3D drawing user interface
- Little need to learn X3D/VRML syntax

Cons:
- May not support all X3D/VRML features
- May not produce most efficient X3D/VRML

- **A shape generator**

Pros:
- Easy way to generate complex shapes (Fractal mountains, logos, etc.)

Cons:
- Only suitable for narrow set of shapes
- Best used with other software

- **A modeler and format converter**

Pros:
- Very powerful features available
- Can make photo-realistic images too
- Easy to make shapes that are too complex

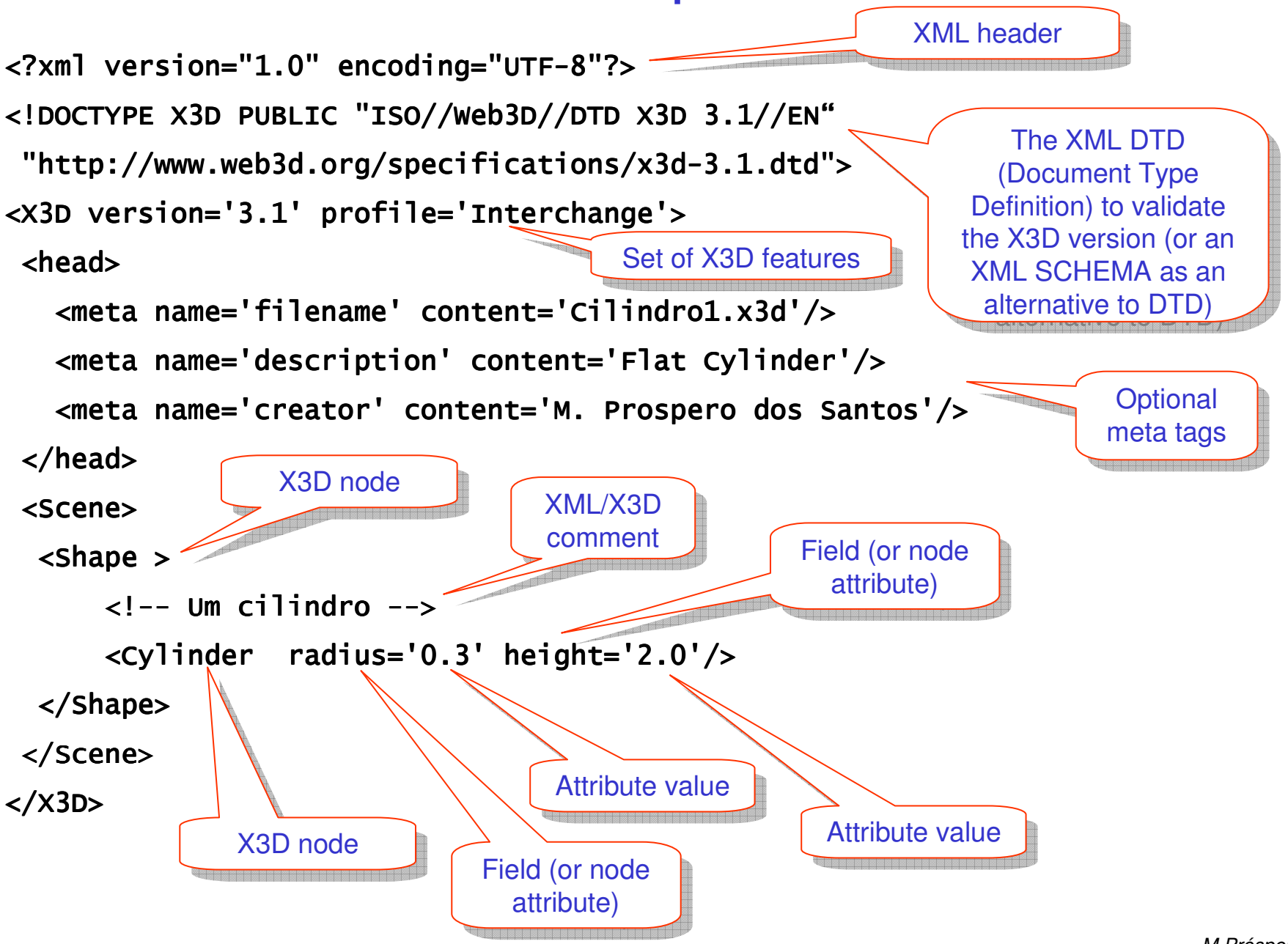
Cons:
- May not support all X3D/VRML features
- Not designed for X3D/VRML
- One-way path from modeler into X3D/VRML

- **An integrated development environment (IDE)**

Pros:
- X3D/VRML syntax validation and support
- Detailed control of world efficiency

Cons:
- Hard to author very complex 3D shapes

An X3D sample file



… And the equivalent VRML code

```
#VRML V2.0 utf8
# Flat Cylinder
#Autor: M. Prospero dos Santos

Shape {
       geometry
            Cylinder { radius .3 height 2 }    # Um cilindro
}
```

Notes about the past:

- The first version of VRML was specified in November 1994 (known as VRML1 or VRML 1.0).
- In 1997, a new version of the format was finalized, as VRML97 (also known as VRML2 or VRML 2.0), and became an ISO standard.

The structure of an X3D file

- XML elements correspond to **X3D nodes**
- XML attributes correspond to **X3D fields**

Nodes . . . . . description of scene content (shapes, lights, sounds, etc.)

Fields . . . . . node attributes you can change

Values . . . . . attribute values

An X3D file may contain any number of node elements.

Nodes may serve as fields to other X3D nodes.

Fields and values specify node attributes.

Fields are optional and given in any order (default value used if field not given).

Every node has:

- A *node type* (Shape , Cylinder , etc.)
- Zero or more fields

Every field has:

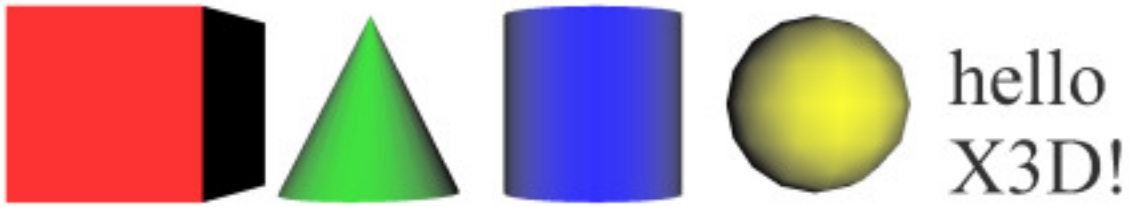
- A field name
- A *data type* (float, int, etc.)
- A default value

Shapes

Shapes are the building blocks of an X3D/VRML world.

Primitive Shapes are standard building blocks:

- **Box**
- **Cone**
- **Cylinder**
- **Sphere**
- **Text**



By default, all shapes are built at the center of the world.

Geometric Transformations

A **Transform node** enables you to

- **Scale** shapes
- **Rotate** shapes
- **Translate** (position) shapes

Note: According to the specification, scale values shall be greater than zero.

Abstract functional specification:

```
Transform : X3DGroupingNode {
  MFNode      [in]      addChildren              [X3DChildNode]
  MFNode      [in]      removeChildren           [X3DChildNode]
  SFVec3f     [in,out]  center           0 0 0   (-∞,∞)
  MFNode      [in,out]  children         []      [X3DChildNode]
  SFNode      [in,out]  metadata         NULL    [X3DMetadataObject]
  SFRotation  [in,out]  rotation         0 0 1 0 [-1,1] or (-∞,∞)
  SFVec3f     [in,out]  scale            1 1 1   (0, ∞)
  SFRotation  [in,out]  scaleOrientation 0 0 1 0 [-1,1] or (-∞,∞)
  SFVec3f     [in,out]  translation      0 0 0   (-∞,∞)
  SFVec3f     []        bboxCenter       0 0 0   (-∞,∞)
  SFVec3f     []        bboxSize         -1 -1 -1 [0,∞) or -1 -1 -1
}
```

Field types

Field names

Default values

Node Specification

Field type (prefix)

`SF...` → one value of the specified data type
`MF...` → a multiple-valued field (`[]` are used)

Interface type (it controls what access the other nodes have to the contents of the field)

`[in]` → write access
`[out]` → read access
`[]` → no access
`[in,out]` → read and write access for all nodes

Another node specification:

```
TimeSensor : X3DTimeDependentNode, X3DSensorNode {
  SFTime  [in,out] cycleInterval    1      (0,∞)
  SFBool  [in,out] enabled          TRUE
  SFBool  [in,out] loop             FALSE
  SFNode  [in,out] metadata         NULL   [X3DMetadataObject]
  SFTime  [in,out] pauseTime        0      (-∞,∞)
  SFTime  [in,out] resumeTime       0
  SFTime  [in,out] startTime        0      (-∞,∞)
  SFTime  [in,out] stopTime         0      (-∞,∞)
  SFTime  [out]    cycleTime
  SFTime  [out]    elapsedTime
  SFFloat [out]    fraction_changed
  SFBool  [out]    isActive
  SFBool  [out]    isPaused
  SFTime  [out]    time
}
```

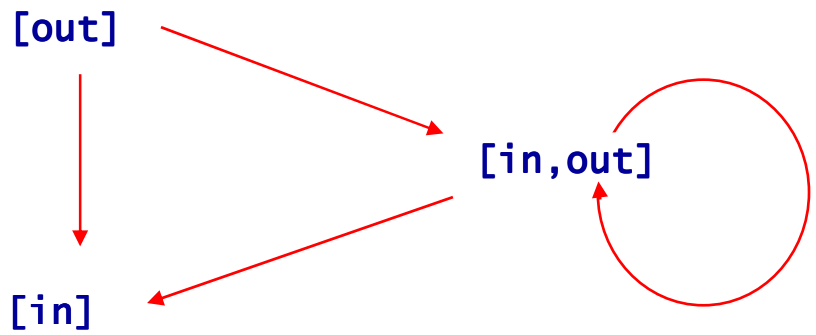
Animation Basics

Animation is created by a sequence of events.
One event can generate any number of other events.

Connecting the output of one node to the input of another:

```
<ROUTE fromField=OutFieldName fromNode=Node1
       toField=InFieldName toNode=Node2 />
```

Admissible connections:

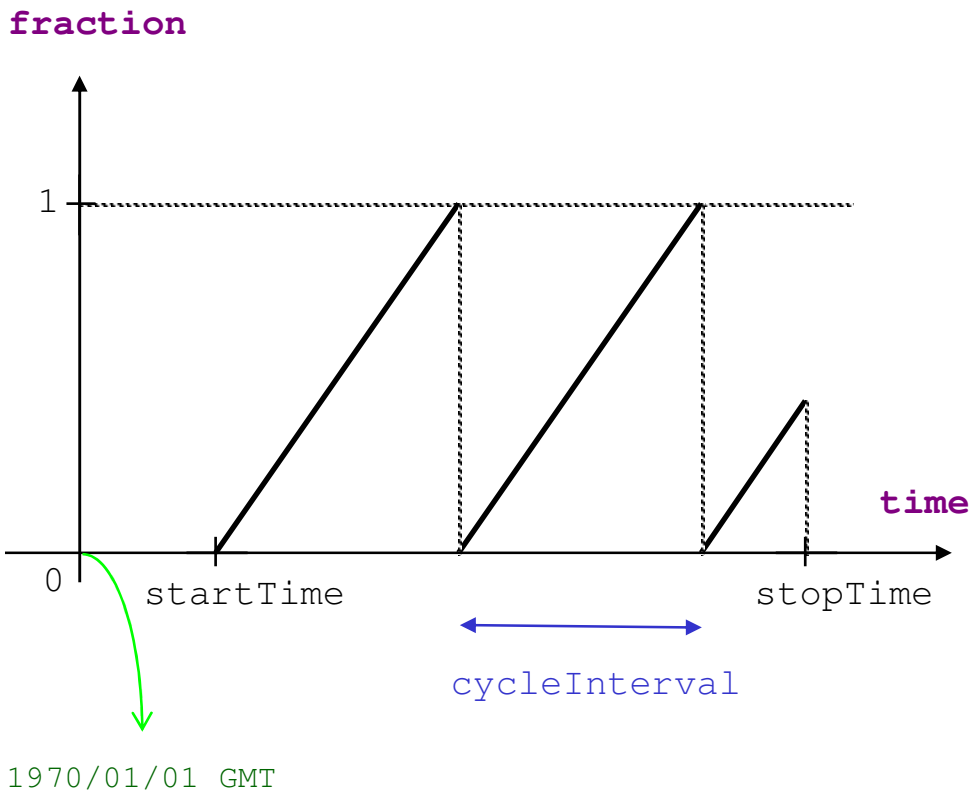


Question: What causes the first event?
Answer: An external influence, e.g.:

- the passing of time
- a user action

There is a lot of control over how events are generated !

The passing of Time…



(see **TimeSensor** node)

Interpolators

```
PositionInterpolator : X3DInterpolatorNode {
  SFFloat [in]     set_fraction       (-∞,∞)
  MFFloat [in,out] key            []  (-∞,∞)
  MFVec3f [in,out] keyValue       []  (-∞,∞)
  SFNode  [in,out] metadata       NULL [X3DMetadataObject]
  SFVec3f [out]    value_changed
}

OrientationInterpolator : X3DInterpolatorNode {
  SFFloat    [in]     set_fraction       (-∞,∞)
  MFFloat    [in,out] key            []  (-∞,∞)
  MFRotation [in,out] keyValue       []  [-1,1] or (-∞,∞)
  SFNode     [in,out] metadata       NULL [X3DMetadataObject]
  SFRotation [out]    value_changed
}

ScalarInterpolator : X3DInterpolatorNode {
  SFFloat [in]     set_fraction       (-∞,∞)
  MFFloat [in,out] key            []  (-∞,∞)
  MFFloat [in,out] keyValue       []  (-∞,∞)
  SFNode  [in,out] metadata       NULL [X3DMetadataObject]
  SFFloat [out]    value_changed
}
```

X3D Baseline Profiles

Interchange is the basic profile for communicating between applications. It support geometry, texturing, basic lighting, and animation.

Interactive enables basic interaction with a 3D environment by adding various sensor nodes for user navigation and interaction (e.g., PlaneSensor, TouchSensor, etc.), enhanced timing, and additional lighting (Spotlight, PointLight).

Immersive enables full 3D graphics and interaction, including audio support, collision, fog, and scripting.

Full includes all defined nodes including NURBS, H-Anim and GeoSpatial components.