VRML 2.0 Summary

DAN OLTEANU

INSTITUT FÜR INFORMATIK, LMU MÜNCHEN

5.02.2001

VRML 2.0 Sections

- Building Shapes
  - ➜ simple shapes (Box, Cone, Cylinder, Sphere, Text)
  - ➜ complex shapes (Point, IndexedLine, IndexedFace) Set
  - ➜ grids and extruded shapes (ElevationGrid, Extrusion)
- Animation - Events, Sensors, Scripts, Interpolators
  - ➜ environmental - (Time, Visibility, Proximity, Collision) Sensors
  - ➜ drag - (Touch, Sphere, Cylinder, Plan) Sensors
  - ➜ (Scalar, Color, Coordinate, Position, Orientation, Normal) Interpolators
- Other features
  - ➜ Lighting and Shading (Point, Directional, Spot) Light
  - ➜ Reusing, Grouping, Linking (DEF/USE, Group, Transform, Switch, Anchor)
  - ➜ Controlling the **L**evel **O**f **D**etail
  - ➜ Creating New Node Types and Scripts

Simple Shapes

`characteristics of a shape: geometry and appearance`

- Appearance
  - ➜ Material - ambientIntensity, (diffuse, emissive, specular) color, transparency, shininess
  - ➜ Texture - (Image, Pixel, Movie) Texture
- Geometry
  - ➜ Box - size
  - ➜ Cone - bottomRadius, height, side, bottom
  - ➜ Cylinder - radius, height, side, top, bottom
  - ➜ Sphere - radius

Complex Shapes

- PointSet - collection of points
- IndexedLineSet, IndexedFaceSet - polylines, polygons
  - ➜ indexed usage from a set of items
  - ➜ direction of normals for faces influences the appearance
  - ➜ planar/non- , concave/convex (no inlets, caves)faces
  - ➜ solid, smoothly shading shapes by providing crease angles
- Elevation Grids
  - ➜ efficiently building faces for a terrain
  - ➜ smooth surfaces usually generated automatically
- Extruded Shapes - cross sections ans spines
  - ➜ mimics the real-world extrusion process
  - ➜ 2-D outline swept along an oriented 3-D curve
  - ➜ controlling the beginning and the end of shapes - **caps**

Animation - Events

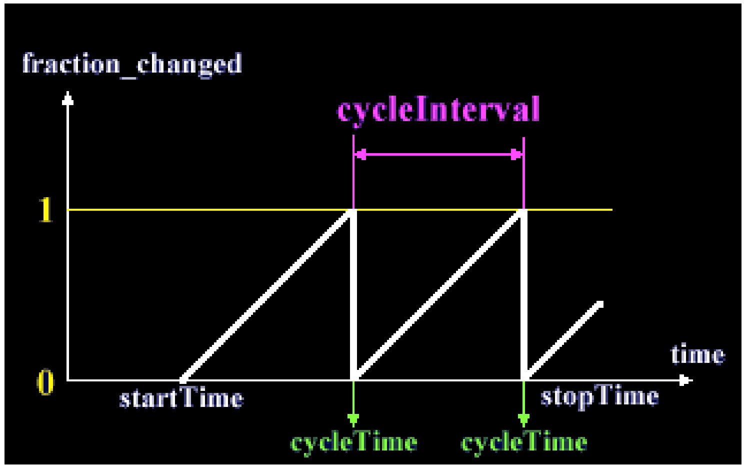
- kind of events - **In**s, **Out**s, **InOut**s
- event = message itself + timestamp (used only internally)
- event cascade = events with the same timestamp
- fan-in versus fan-out
- initial/intermediate events
- routing events of the *same* type

TIMESENSOR

eventOuts

cycleTime, isActive, time, fractionChanged

cycleInterval, enabled, loop, startTime, edTime



Environmental Sensors

- Visibility sensor - optimizing your world

  1. stopping the animations that are outside the users field of vision
  2. eventOuts: enter/exitTime, size, center, enabled, isActive

- Proximity sensor - sensible to your proximity

  1. inside its active area, will generate events whenever its position/orientation change
  2. eventOuts: isActive, position, orientation, enter/exitTime

- Collision detection

  1. appearance of the object (solid if detection activated)
  2. providing proxy, INSTEAD OF the geometry of its children, to collide with – optimization reasons
  3. eventOuts: collideTime, collide

Drag Sensors

- Touch Sensor

  1. detecting mouse interaction with its sibling geometry
  2. eventOuts: touchTime, hitNormal, hitPoint, isOver; isActive, enabled

- Sphere Sensor

  1. allows rotating the object around the local origin of the sensor
  2. eventOuts: rotation, trackPoint; auto-/offset, isActive, enabled

- Cylinder Sensor

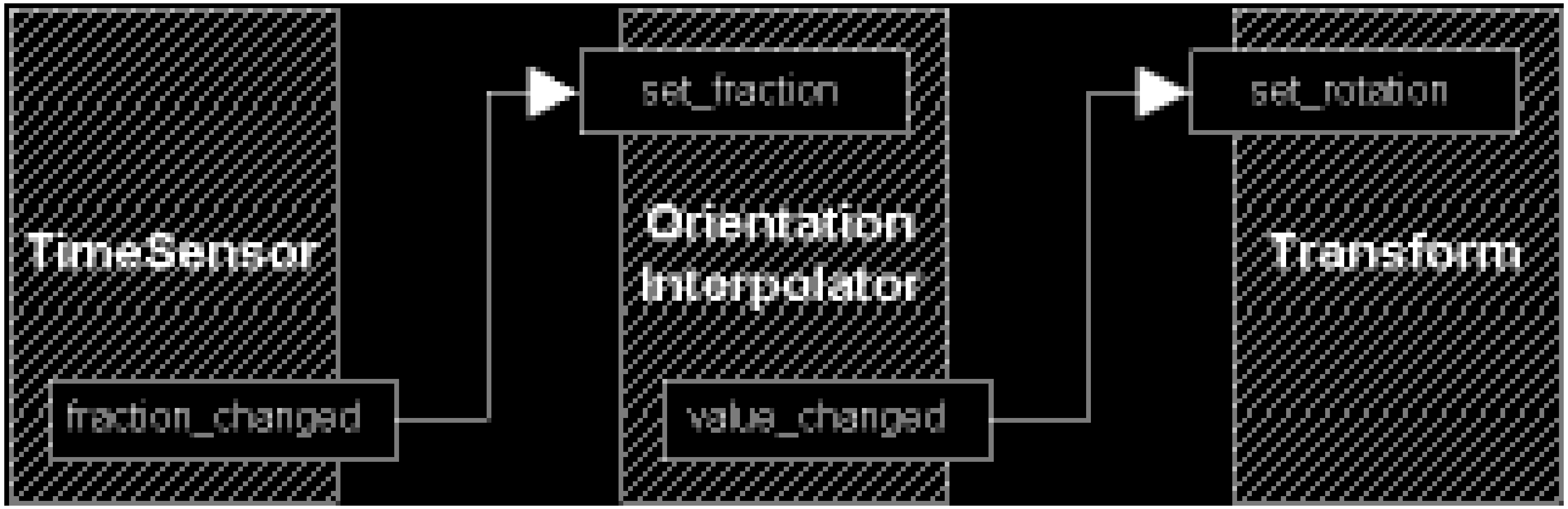
  1. like the previous sensor, but constrained to rotate only around its local y-axis with angle limitators
  2. eventOuts: rotation, trackPoint, max/min/diskAngle
  3. 2 behaviours modelled by diskAngle: disk-like pin and rollingpin-like roll

- Plane Sensor

  1. provides dragging in a XY plane with 2-D coordinate limitators
  2. eventOuts: translation, trackPoint, max/minPosition

InBetweener - Interpolators

1. used to change a certain value over time
2. take timing signals from sensors and perform a linear interpolation between sets of values (keyValues)
3. types: Color, Coordinate, Normal, Orientation, Position, ScalarInterpolator
4. eventIns: fraction
5. eventOuts: value
6. exposedField: key, keyValue



CONTROLLING SHADING

- Math backgrounds

```
1. normal - way of indicating the orientation of a face
2. determining normal direction - right-hand rule
3. computing normals V(x,y,z), W(m,n,p) -> N(y*p-z*n,z*m-x*p,x*n-y*m)
4. unit vectors (magnitude = 1)
```

- Algorithm for computing normals

```
1. compute normals for each coord. of the shape;
2. look at adjacent faces on your shape - there are multiple
normals, i.e. one for each face
3. compute the average normal for the previous step and use it as the
smooth shading coord. normals for your shape
```

- What is *creaseAngle*?

```
a way of automatically computing the normals, if you do not specify
your own
```

- VRML Specified Support

```
1. Normal Node
2. normalPerVErtex and normal index attributes
3. NormalInterpolator Node
```

Lighting your Worlds

- Types

  default: browser's headLight

  ➜ PointLight

    1. it emanates light in a radial pattern in all directions
    2. attenuation parameters: radius, attenuation (3 control fields)

  ➜ DirectionalLight

    1. light aimed from infinitely far away so that all of its rays are parallel and point in the same direction
    2. no attenuation supported

  ➜ SpotLight

    1. light placed at a location aimed in a specific direction
    2. attenuation: extra cutOffAngle and beamWidth

- Effects

  1. no form or shape
  2. influence the appearance of an object, its shading
  3. light attenuation – how illumination gradually drops off with distance from the light source – for Point/SpotLight

Grouping in VRML

- Goals

  `ensures a close connection between various objects, allowing the inheritance mechanism between parent-children and the same behaviour wrt. various sensors, transformations between siblings`

- Children ordering
  - ➜ the order matters: LOD, Switch
  - ➜ the order doesn't matter: Group, Transform, Anchor, Inline..

Optimization issues - Levels Of Detail

- built-in mechanism, syntactically emulated using *Switch* and *Sensing Viewer's Proximity*
- encapsulates different detail versions of the same object
- different meanings in interpreting the children's order than other grouping nodes

CREATING NEW NODE TYPES

1. provided support: already 50 different node types
2. ease-made extensions, providing interface and body for the new node
3. the new node may have geometry and/or appearance

- Node Interface

```
PROTO aNode [
 field fieldType fieldName defaultValue
] { .node body. }
where fietd = {exposedField, eventIn, eventOut, field}
```

- Node Body

  contains a list of any VRML nodes, routes to wire these nodes

- Creating multiple prototypes

  1. nesting definitions - valability domain
  2. external libraries, each of which having multiple nodes inside

- Connectors

  1. IS = connection between an interface field in a PROTO definition and a field in the definition's node body

PROGRAMMING EXTENSIONS - SCRIPTS

- Goals
  - ➜ allowing the creation of nodes with complex behaviour
  - ➜ may describe custom-made sensors and interpolators
  - ➜ associates local/remove Java/-Script/.. code with VRML nodes
  - ➜ enriched control mechanisms -optimization issues
    - ➜ *evaluation* control- every time a value is received/postponing evaluation
    - ➜ *output* control - control over the whole/part of the world
- delivering events
  - ➜ servlet behaviour
  - ➜ three types of activities: intialization, shutdown, event receipt