Modelica3D
Platform Independent Simulation Visualization

Christoph Höger
Technische Universität Berlin
Fraunhofer FIRST
Motivation - Goal

Dymola MultiBody Visualization
Motivation - Goal

Goal: Get those fancy 3D graphics standardized

- OpenSceneGraph
- blender
- Modelica
Modelica3D Design

Modelica3D is . . .

- **Platform-independent:**
  
  *Modelica3D can be used with OpenModelica, Dymola, . . .*  
  *Modelica3D uses only standardized techniques.*

- **Extensible:**

  *Modelica3D can be extended with new functionality (new shapes, different visualization tools etc.)*

- **Leightweight:**

  *Modelica3D does not depend on any Modelica or C libraries. It does not contain large layers of abstraction or multitudes of features.*
Modelica3D Design

Modelica3D is . . .

► Platform-independent:

*Modelica3D can be used with OpenModelica, Dymola, . . . .
Modelica3D uses only standardized techniques.*

► Extensible:

*Modelica3D can be extended with new functionality (new shapes, different visualization tools etc.)*

► Lightweight:

*Modelica3D does not depend on any Modelica or C libraries. It does not contain large layers of abstraction or multitudes of features.*
Modelica3D Design

Modelica3D is . . .

- **Platform-independent:**
  
  Modelica3D can be used with OpenModelica, Dymola, . . . .

  Modelica3D uses only standardized techniques.

- **Extensible:**

  Modelica3D can be extended with new functionality (new shapes, different visualization tools etc.)

- **Lightweight:**

  Modelica3D does not depend on any Modelica or C libraries. It does not contain large layers of abstraction or multitudes of features.
Modelica3D is . . .

- Platform-independent:
  Modelica3D can be used with OpenModelica, Dymola, . . . .
  Modelica3D uses only standardized techniques.

- Extensible:
  Modelica3D can be extended with new functionality (new shapes,
  different visualization tools etc.)

- Leightweight:
  Modelica3D does not depend on any Modelica or C libraries. It does
  not contain large layers of abstraction or multitudes of features.
Modelica3D Design II

Modelica3D is not . . .

- Declarative:
  
  Since Modelica3D is a small library it does not contain 3D-connectors or equation support or static graphics.
Modelica3D is not . . .

- Declarative:
  
  Since Modelica3D is a small library it does not contain 3D-connectors or equation support or static graphics.
Architecture
Modelica3D Architecture

- Client/Server approach
- Client: Simulation enhanced with Modelica3D
- Server: Visualization tool (e.g. Blender, 3DSMax, etc.)
- Unidirectional communication (via TCP/IP)
- Physics drive visualization
- Visualization can be ignored, stored, rendered
Modelica3D Client Architecture

Model

links against

libmodelica3d.so

calls

sends

contains

contains

Physics

defines

GFX

3D package
Modelica3D Server Architecture

- **Vis. Tool** provides **3D API**
- **3D package** reads **Unpacker**
- **3D API** is extended by another component
- **Unpacker** delegates to another component
API - Objects & Operations
Modelica3D Shapes
3D-shapes can be *instantiated*

```model
model Foo
    import ModelicaServices.Modelica3D;
    Modelica3D.Cylinder cyl;
end Foo;
```
3D-shapes are *external objects* (can be allocated directly)
Shape definition is an *operation*.

class Cylinder "A 3D cylinder object"
  String  name = "";
  ExtID   id = ExtID(name);
  ...
  when initial() then
    dummy := InitCylinder(id, diameter, length);
  end when;
end Cylinder;
Modelica3D Operations

- **group**
- **rotate**
- **move**
- **scale**
Modelica3D API - Procedure Pattern

Modelica3D API Convention

Operation | Object ID | Parameters | Frame #
--- | --- | --- | ---

Modelica3D.Translate(cylinder.id, 5.0, 0.0, 0.0, update3d.frame):

- Most operations follow this common pattern.
Operations require a frame to take effect.
Frames and simulation-time need to be synchronized
Solution Update3D object and when-clause.

```modelica
class Update3D "Define communication and update interval"
    parameter Integer framerate = 30;
    parameter Modelica.SIunits.Time updateInterval = 1 / framerate;
    output Boolean send;
    output Integer frame;

    equation
        send = sample(1e-08, updateInterval);

    algorithm
        when send then
            frame := integer( time/updateInterval + 1); // First frame is 1, not 0;
            setFrameTime(frame, time);
        end when;
end Update3D;
```
Application Example: MultiBody
MultiBody

- State-of-the-art in Modelica visualization
- Supports shapes, materials, transformations
- Has a global state model bundled in Modelica.Utilities.Internal.PartialShape
- Modelica.Mechanics.MultiBody.Pendulum has 1500 variables
- Idea: Extend PartialShape to be extended by Modelica3D calls
- Problem: Conversion between state and Modelica3D nontrivial
- Hit some omc bugs
Demo Video
Interested?

- Mail me for an evaluation license (christoph.hoeger AT tu-berlin.de).
- Create an account at https://mlcontrol.uebb.tu-berlin.de/redmine/
- Compile Modelica3D
- Get Blender / OSG
- Visualize the Demos
Future Development

- Publish under free license
- Complete MultiBody Support
- Implement Utility Blocks
- More renderers
Thank you for your attention!

Any Questions?