Interactive 3D Graphics for Tcl/Tk
Oliver Kersting and Jürgen Döllner

3rd European Tcl/Tk User Meeting
June 2002, Munich

Overview

1. Interactive 3D Graphics
2. Interactive Virtual Rendering System
3. API Mapping Technique
4. Developing 3D Applications with iVRS
5. Conclusions
1. Interactive 3D Graphics

Applications of interactive 3D graphics
- Information Visualization
- Scientific Visualization
- CAD/CAM
- Entertainment and Gaming
- Education

Elements of interactive 3D graphics
- Rendering of 3D scenes in real-time
- Interaction with 3D objects and 3D scenes
- Animation of 3D objects and 3D scenes

Developing interactive 3D graphics applications
- Programming based on low-level libraries, e.g., OpenGL
- Programming based on higher-level toolkits, e.g., OpenInventor, Java3D

Characteristics:
- System programming languages
- High performance
- API with large number of data structures, functions, or classes
- Strong typing
1. Interactive 3D Graphics

Difficulties developing 3D Applications

- **Programming** and **Configuring** of 3D applications
  
  How to modify 3D scenes?
  How to experiment with features?

  ➔ Every access by system programming language requires compile-link cycles, which increase development time

- **Exploring** and understanding of 3D graphics libraries
  
  How to find features?
  Which function do I need? …

  ➔ Difficult to find appropriate functionality in large and complex APIs

Our Solution

- Apply a high-level object-oriented 3D graphics library
- Map its C++ API and meta information to Tcl

- **Program and configure** 3D graphics applications interactively using the Tcl interpreter
- **Explore** API by Tcl commands
2. Interactive Virtual Rendering System

Virtual Rendering System (VRS)

**General-purpose 3D graphics library**
- Support for 3D modeling, interaction, and animation
- Scene graph
- Rendering based on OpenGL

**Implementation**
- Object-oriented
- Written in C++
2. Interactive Virtual Rendering System

Virtual Rendering System (VRS)

- **Advanced real-time rendering techniques**
  - Shadows
  - Reflections
  - Bump mapping
  - Multi-texturing

- **IO support**
  - Image: bmp, ppm, jpeg, tiff …
  - Video: avi, mpeg

- **2D Imaging**
  - Image manipulation
  - Convolution filtering

- **Support for additional rendering systems**
  - BMRT (RenderMan)
  - POVRay

---

2. Interactive Virtual Rendering System

**VRS Core Elements**

- **Shapes**
  - sphere, cylinder, point, line,
    level-of-detail mesh, …

- **Graphics Attributes**
  - color, material, texture,
    light sources, …

- **Transformations**
  - rotation, scaling, translation,
    billboarding …

- **Nodes**
  - container objects
    build scene graphs
2. Interactive Virtual Rendering System

Observations

- Manipulation of scene graphs **occurs frequently** during 3D application development
- Manipulation of scene graphs **implies recompilation and linking**

» Scene graph manipulation is a time-critical aspect in developing 3D graphics applications
» How can we speed up developing process?

**Interactive Virtual Rendering System**

= Easily program 3D graphics by scripting, thereby doing time-critical operations in C++
+ Map VRS API to corresponding Tcl commands
+ Create, manipulate, destroy VRS objects by Tcl

» Interactive 3D application development
access to class and API reflection information, reconfiguration of all objects at run-time

» **No loss of rendering performance**
rendering as time-critical part is executed at C++ level
2. Interactive Virtual Rendering System

Example: C++ API mapped to Tcl

VRS/C++

```
Sphere* mysphere = new Sphere(12);
mysphere->setRadius(15);
deleter mysphere
```

iVRS/Tcl

```
set mysphere [new Sphere 12]
$mysphere setRadius 15
delete $mysphere
```
3. API Mapping Technique

Major Steps of the Mapping Process
- Analyze C++ API
- Generate C++ wrapper code
- Compile C++ wrapper code
- Build Tcl extension package

Mapping Features
- Static, virtual, and overloaded methods
- Default arguments
- Enumerations
- Template classes
- Reference counting

⇒ Wrapper classes and method tables
3. API Mapping Technique – Wrapper Class

iVRS Wrapper Class (Implementation Detail)
- Reflects interface of a VRS class with wrapper methods which exclusively use string arguments
- A wrapper method converts incoming string arguments to original types, completes missing arguments with default values, and calls the wrapped method

```
Wrapped Class

wrappedA
  +getValue():double
  +setValue(v:double,b:bool):void
  +modified():void

Wrapper Class

AWrap
  +obj:A *
  +getValue(argc:int,argv:char * *):char *
  +_setValue_double_bool(argc:int,argv:char * *):char *
  +_modified(argc:int,argv:char * *):char *
```

Wrapped Method

[Diagram showing method pointers and arguments]

```
Method Pointer

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Arguments</th>
<th>Min</th>
<th>Max</th>
<th>Method Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;A&quot;</td>
<td>&quot;double&quot;</td>
<td>1</td>
<td>1</td>
<td>AMrap::_A_double</td>
</tr>
<tr>
<td>&quot;setValue&quot;</td>
<td>&quot;double bool&quot;</td>
<td>1</td>
<td>2</td>
<td>AMrap::_setValue_double_bool</td>
</tr>
<tr>
<td>&quot;getValue&quot;</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>AMrap::_getValue</td>
</tr>
<tr>
<td>&quot;modified&quot;</td>
<td>&quot;&quot;</td>
<td>0</td>
<td>0</td>
<td>AMrap::_modified</td>
</tr>
</tbody>
</table>
```

→ Enables iVRS to call polymorph methods, methods using default values and overloaded methods
4. Developing 3D Applications with iVRS

4. Examples
4. Examples - 3D Object Viewer

```tcl
package require iVRS

set myCanvas [new TclCanvas .view 400 400]
pack .view

set myScene [new SceneThing]
set myCamera [new Camera {0 -2 -2} {0 0 0} 60]
$myScene append $myCamera

set distantlight [new DistantLight]
$myScene append $distantlight

set my3ds [ObjectLoader readFile dragon.3ds]
$myScene append $my3ds

$myCanvas append $myScene
$myCanvas append [new TrackBall $my3ds]
```

4. Examples – iVRS IDE

iVRS Integrated Development Environment

Meta information at run-time
- Base class and child classes
- Methods including complete signature
- Enumerations
- Instantiated objects
- Object relationships

➤ Automated GUI components for VRS objects
➤ Integrated help system
4. Examples – LandExplorer

LandExplorer: 3D Map System based on iVRS

5. Conclusions
5. Conclusions

iVRS
- Allows developers to **program** and **configure** interactive 3D graphics applications interactively at run-time
- Allows developers to **explore** the complete API interactively
- Supports **platform-independent** 3D graphics application development
- Facilitates **rapid prototyping**
- Offers real-time rendering for scripting languages without any remarkable loss of **performance**

## Future Work and License

### Future Work
- Add C++ comments to iVRS meta information
- Add VRS namespace in Tcl
- Improve error messaging
- Support for additional scripting languages

### License
iVRS is Open Source Software
GNU Lesser General Public License
Thank you.

www.vrs3d.org