Interactive 3D Graphics for Tcl/Tk

Oliver Kersting and Jürgen Döllner

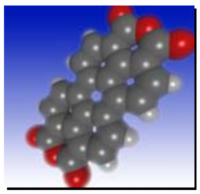
3rd European Tcl/Tk User Meeting
June 2002, Munich

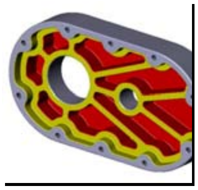
HASSO-PLATTNER-INSTITUT
for Software Systems Engineering
at the University of Potsdam

Overview

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

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

⇩

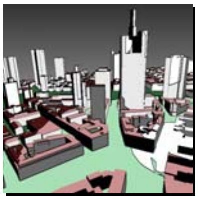
- **Program and configure** 3D graphics applications interactively using the Tcl interpreter
- **Explore** API by Tcl commands

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++



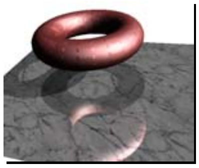
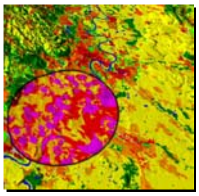
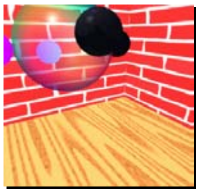
Virtual Rendering System (VRS)

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

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

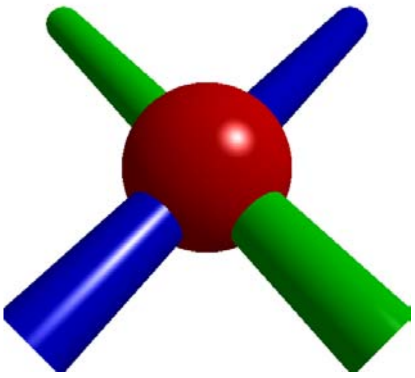
2D Imaging
- Image manipulation
- Convolution filtering

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

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



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

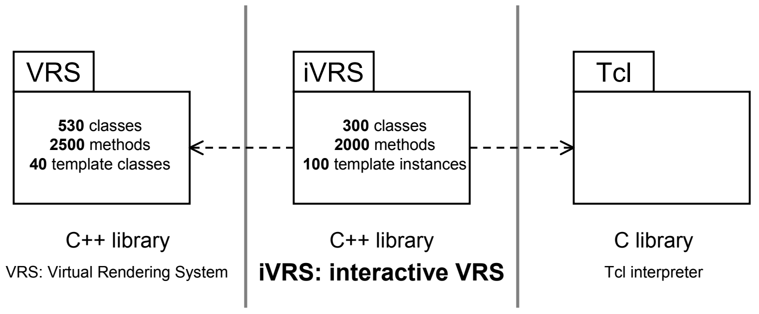
= Easily program 3D graphics by scripting,
thereby doing time-critical operations in C++

+ 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



Example: C++ API mapped to Tcl

VRS/C++

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

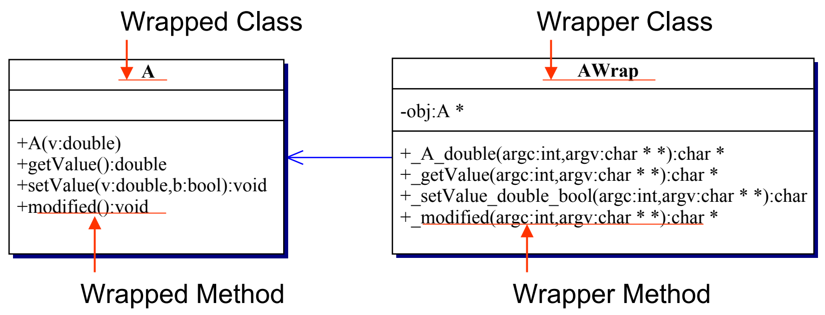
Mapping Features

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

➔ Wrapper classes and method tables

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
Wrapper Class
A
+A(v:double)
+getValue():double
+setValue(v:double,b:bool):void
+modified():void
AWrap
-obj:A *
+_A_double(argc:int,argv:char * *):char *
+_getValue(argc:int,argv:char * *):char *
+_setValue_double_bool(argc:int,argv:char * *):char
+_modified(argc:int,argv:char * *):char *
Wrapped Method
Wrapper Method

iVRS Method Table (Implementation Detail)

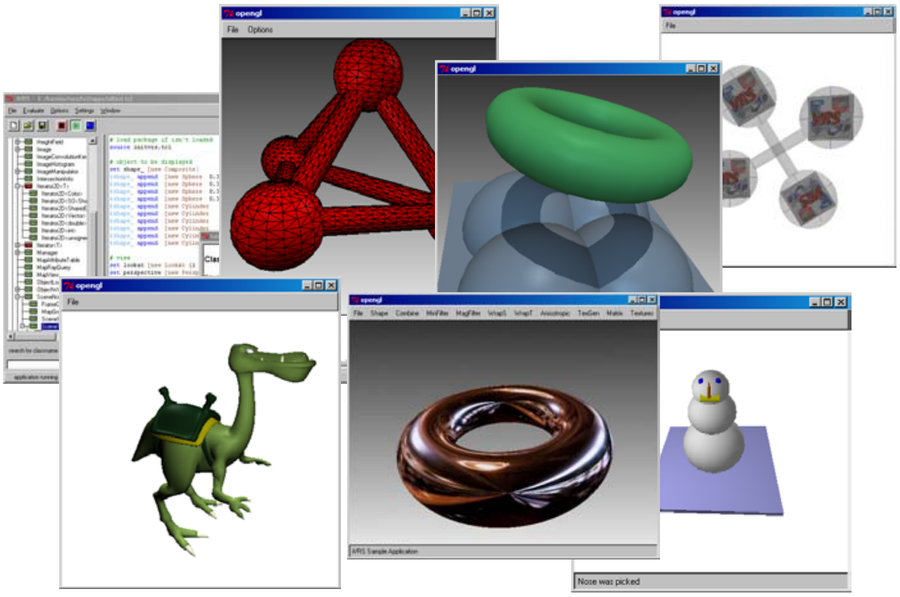
- Stores information about signatures of methods of wrapped classes
- Signature information is required to decide which wrapped method should be called at run-time

Method Name	Arguments	Min	Max	Method Pointer
“A”	“double”	1	1	AWrap::_A_double
“setValue”	“double bool”	1	2	AWrap::_setValue_double_bool
“getValue”	“”	0	0	AWrap::_getValue
“modified”	“”	0	0	AWrap::_modified

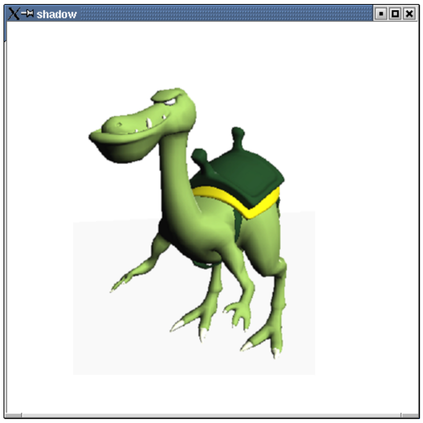
➔ Enables iVRS to call polymorph methods, methods using default values and overloaded methods

4. Developing 3D Applications with iVRS

4. Examples


opengl
File Options
opengl
File
opengl
File
opengl
File Shape Combine MinFilter MagFilter WrapS WrapT Anisotropic TexGen Matrix Textures
iVRS Sample Application
Nose was picked

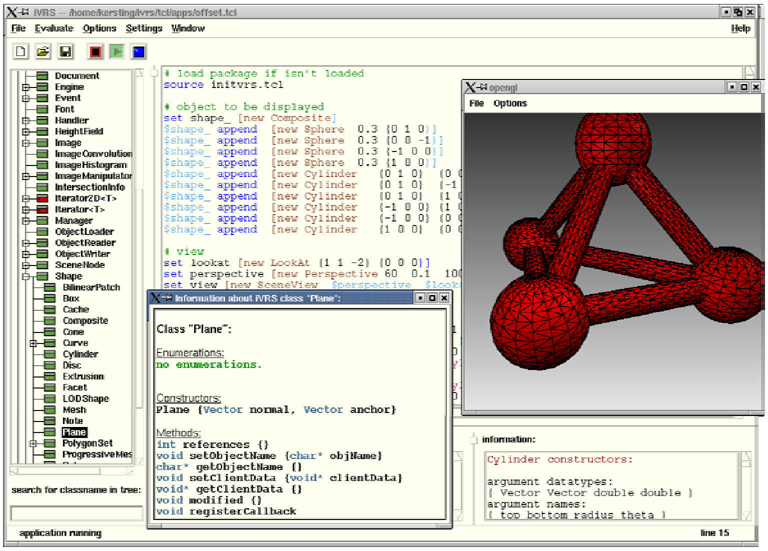
4. Examples - 3D Object Viewer



4. Examples - 3D Object Viewer

```
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]
```

iVRS Integrated Development Environment



```
# load package if isn't loaded
source initvrs.tcl

# object to be displayed
set shape_ [new Composite]
$shape_ append  [new Sphere  0.3 {0 1 0}]
$shape_ append  [new Sphere  0.3 {0 0 -1}]
$shape_ append  [new Sphere  0.3 {-1 0 0}]
$shape_ append  [new Sphere  0.3 {1 0 0}]
$shape_ append  [new Cylinder   {0 1 0}   {0 0
$shape_ append  [new Cylinder   {0 1 0}   {-1
$shape_ append  [new Cylinder   {0 1 0}   {1 0
$shape_ append  [new Cylinder   {-1 0 0}  {1 0
$shape_ append  [new Cylinder   {-1 0 0}  {0 0
$shape_ append  [new Cylinder   {1 0 0}   {0 0

# view
set lookat [new LookAt {1 1 -2} {0 0 0}]
set perspective [new Perspective 60  0.1  100
set view [new SceneView  $perspective  $looka
```

```
Class "Plane":

Enumerations:
no enumerations.

Constructors:
Plane {Vector normal, Vector anchor}

Methods:
int references {}
void setObjectName {char* objName}
char* getObjectName {}
void setClientData {void* clientData}
void* getClientData {}
void modified {}
void registerCallback
```

```
Cylinder constructors:

argument datatypes:
{ Vector Vector double double }
argument names:
{ top bottom radius theta }
```

iVRS Integrated Development Environment

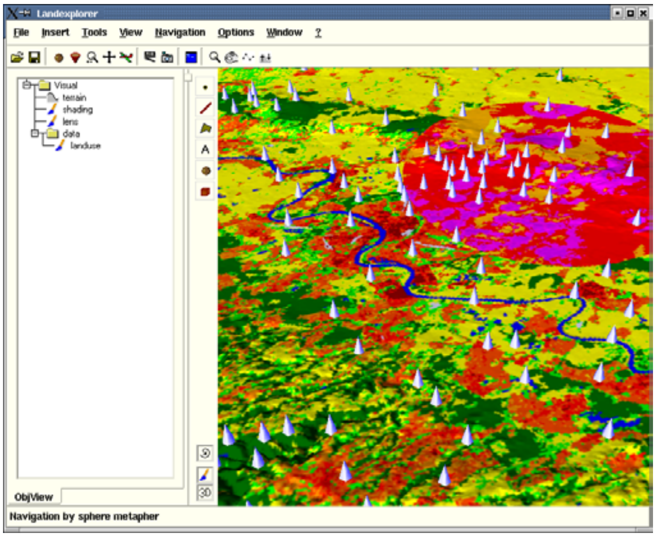
Meta information at run-time

- Base class and child classes
- Methods including complete signature
- Enumerations
- Instantiated objects
- Object relationships

➔ Automated GUI components for VRS objects

➔ Integrated help system

LandExplorer: 3D Map System based on iVRS


Landexplorer
File Insert Tools View Navigation Options Window ?
Visual
terrain
shading
lens
data
landuse
ObjView
Navigation by sphere metapher

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

- 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

HASSO-PLATTNER-INSTITUT
for Software Systems Engineering
at the University of Potsdam