X-Ray: A Photorealistic Rendering System

Ioannis Tsiombikas

nuclear@siggraph.org

Computer graphics

Algorithms to transform mathematical representations of 3D environments to images. The process is called “*rendering*”

Possible surface representations:

Computer graphics

Algorithms to transform mathematical representations of 3D environments to images. The process is called “*rendering*”

Possible surface representations:

- Polyhedral approximation of surfaces.

Computer graphics

Algorithms to transform mathematical representations of 3D environments to images. The process is called "*rendering*"

Possible surface representations:

- Polyhedral approximation of surfaces.
- Mathematical equations describing surfaces (i.e. $x^2 + y^2 + z^2 = r^2$).

Computer graphics

Algorithms to transform mathematical representations of 3D environments to images. The process is called “*rendering*”

Possible surface representations:

- Polyhedral approximation of surfaces.
- Mathematical equations describing surfaces (i.e. $x^2 + y^2 + z^2 = r^2$).
- Volume defined by density values (binary or not) at discrete points in a 3D scalar field (voxels).

Rendering Algorithms

A major distinction in graphics algorithms: real-time vs off-line rendering.

Off-line rendering algorithms, perform complicated and time-consuming calculations, trying to simulate many aspects of the interaction of light with the virtual environment, thus producing images indistinguishable from reality.

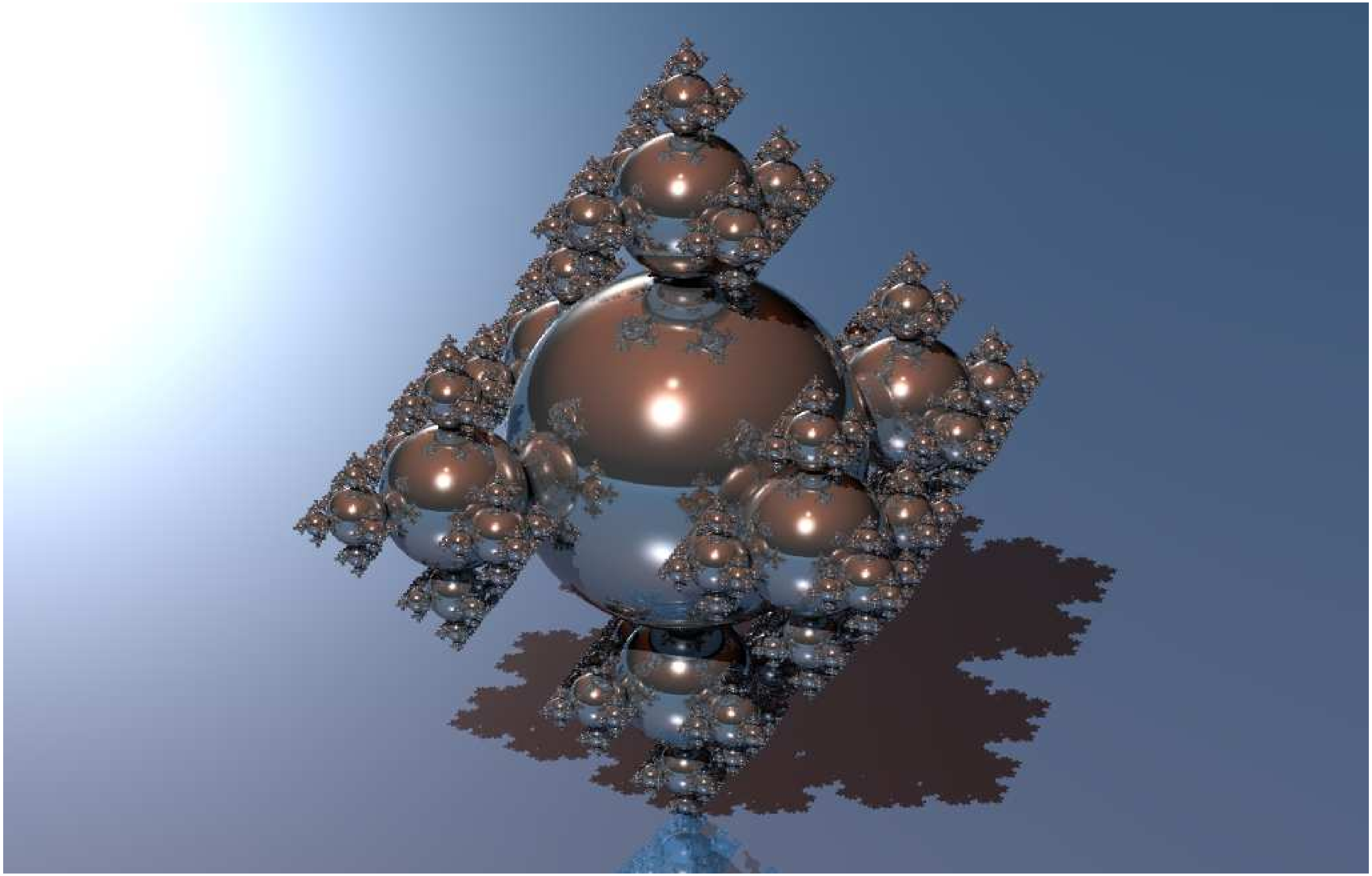
Rendering Algorithms

A major distinction in graphics algorithms: real-time vs off-line rendering.

Off-line rendering algorithms, perform complicated and time-consuming calculations, trying to simulate many aspects of the interaction of light with the virtual environment, thus producing images indistinguishable from reality.

“*The most widely used algorithm?*”

Ray Tracing



About X-Ray

X-Ray is a "*free, cross-platform, programmable, photorealistic renderer*"

About X-Ray

X-Ray is a “*free, cross-platform, programmable, photorealistic renderer*”

- Uses advanced rendering algorithms based on ray tracing.

About X-Ray

X-Ray is a “*free, cross-platform, programmable, photorealistic renderer*”

- Uses advanced rendering algorithms based on ray tracing.
- Lets the user write “*shaders*” to compute illumination and drive the rendering process.

About X-Ray

X-Ray is a “*free, cross-platform, programmable, photorealistic renderer*”

- Uses advanced rendering algorithms based on ray tracing.
- Lets the user write “*shaders*” to compute illumination and drive the rendering process.
- Fully reusable core, in the form of a C++ library.

About X-Ray

X-Ray is a “*free, cross-platform, programmable, photorealistic renderer*”

- Uses advanced rendering algorithms based on ray tracing.
- Lets the user write “*shaders*” to compute illumination and drive the rendering process.
- Fully reusable core, in the form of a C++ library.
- Flexible client/server architecture.

About X-Ray

X-Ray is a “*free, cross-platform, programmable, photorealistic renderer*”

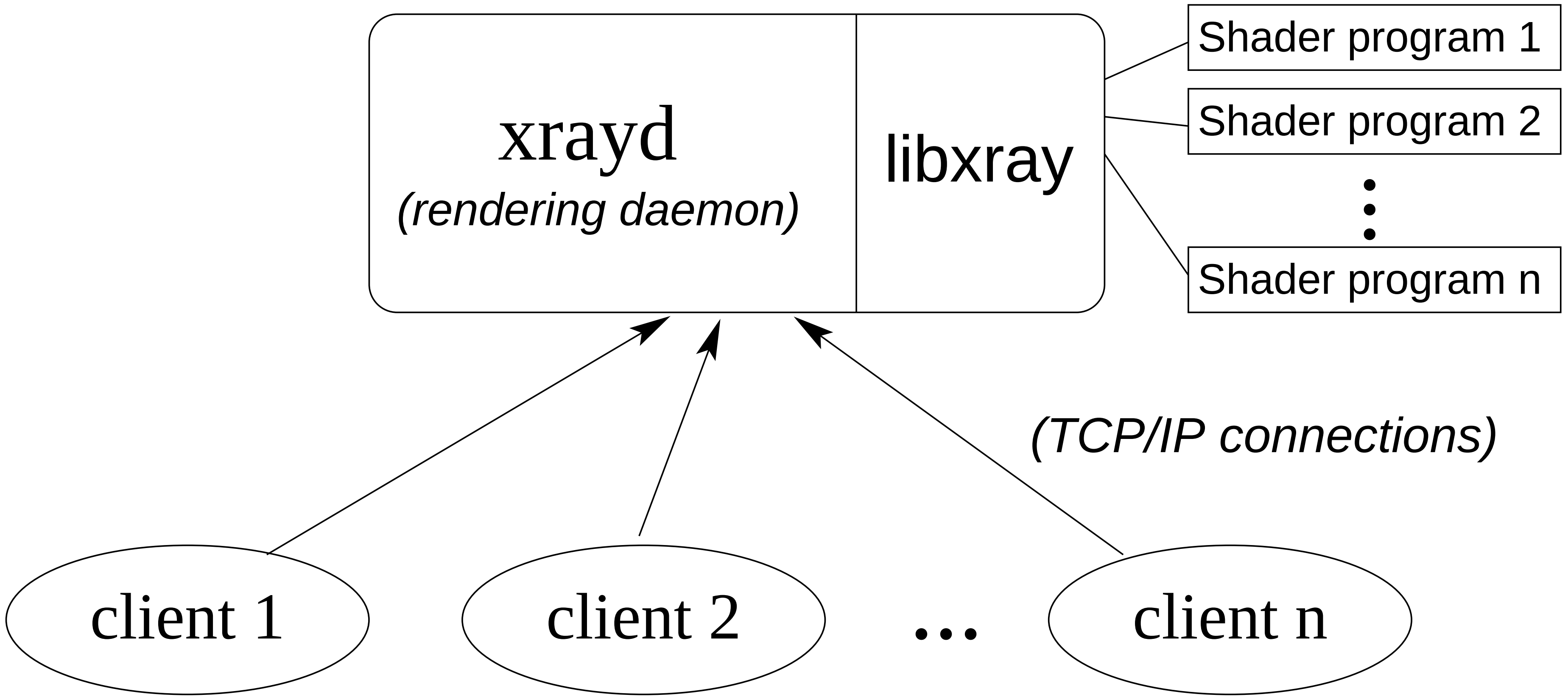
- Uses advanced rendering algorithms based on ray tracing.
- Lets the user write “*shaders*” to compute illumination and drive the rendering process.
- Fully reusable core, in the form of a C++ library.
- Flexible client/server architecture.
- Runs on multiple operating systems and computer architectures.

About X-Ray

X-Ray is a "*free, cross-platform, programmable, photorealistic renderer*"

- Uses advanced rendering algorithms based on ray tracing.
- Lets the user write "*shaders*" to compute illumination and drive the rendering process.
- Fully reusable core, in the form of a C++ library.
- Flexible client/server architecture.
- Runs on multiple operating systems and computer architectures.
- Free Software, released under the GNU General Public License.

X-Ray Architecture

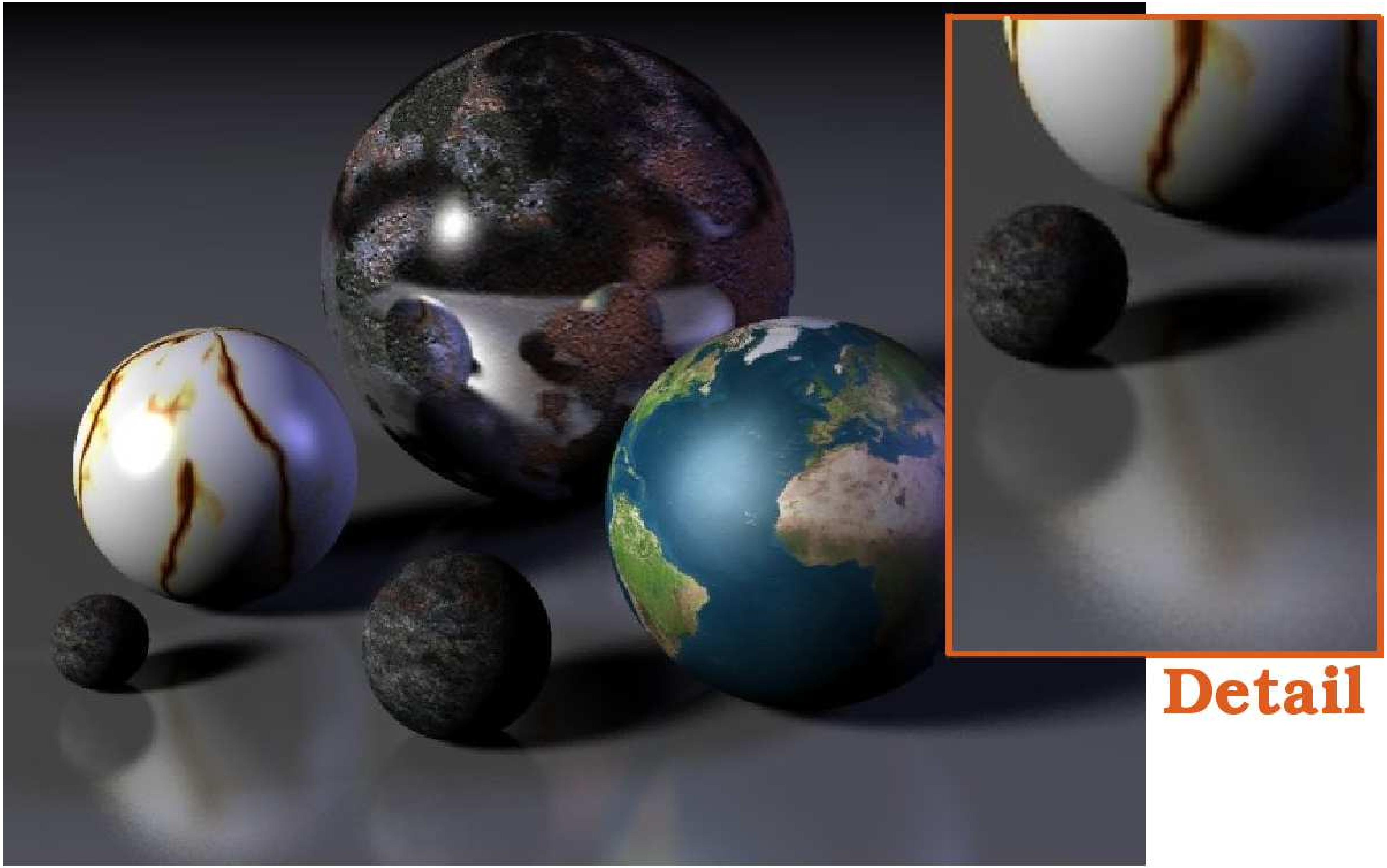


Scene Description

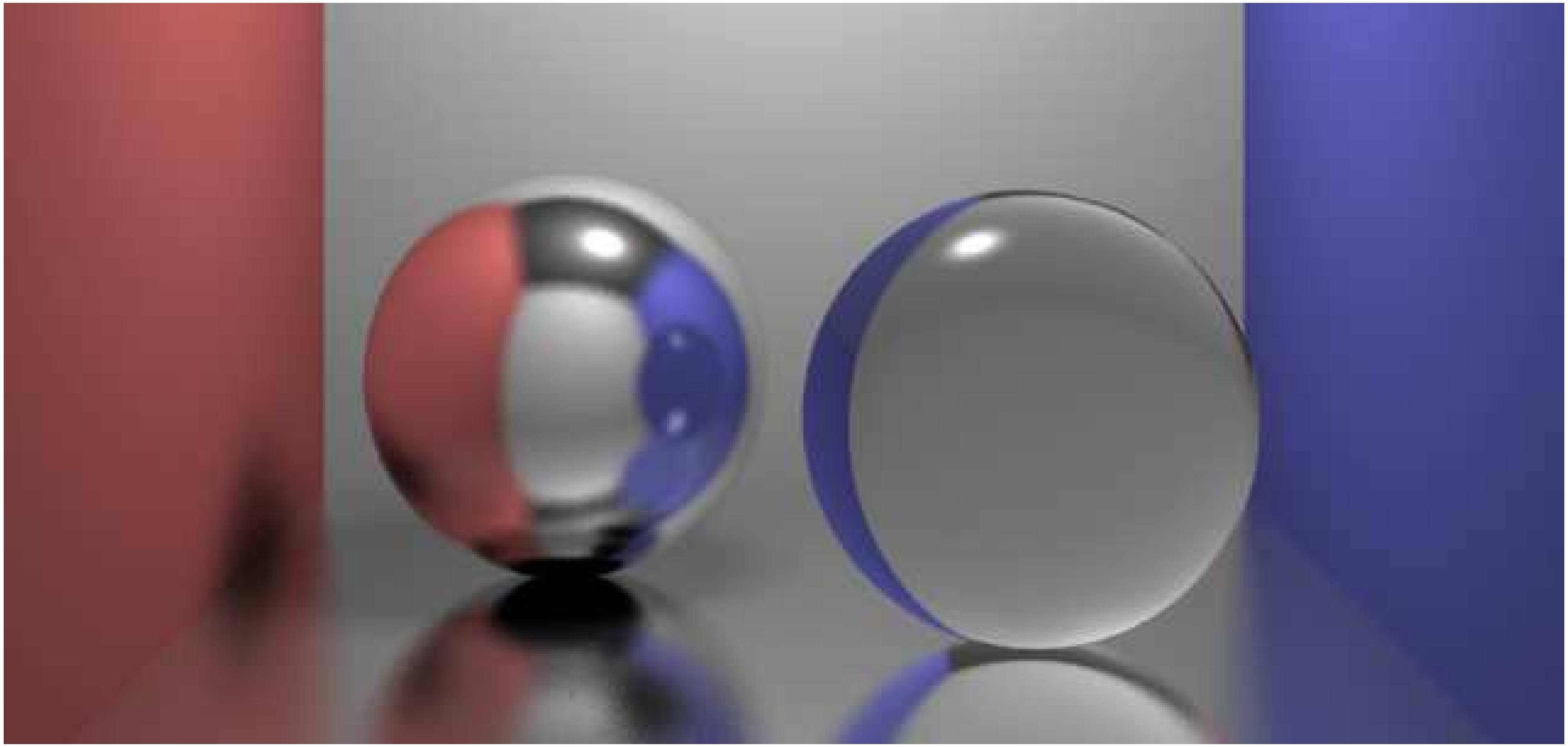
A custom XML file format was designed, as a scene input format for X-Ray.

```
<scene name="example scene">
  <mat-list>
    <material name="Material01" shader="phong.cc"/>
      <attr name="diffuse" val="1, 0.3, 0.1" tex="foo.png"/>
      <attr name="shininess" val="50"/>
    </material>
  </mat-list>
  <object name="ball">
    <matref ref="Material01"/>
    <xform pos="0, 0, 0"/>
    <sphere rad="1"/>
  </object>
  <light name="light01" type="point" pos="-10, 6, -10"/>
  <camera name="cam01" pos="0, 5, -8" target="0, 0, 0" fov="45"/>
</scene>
```

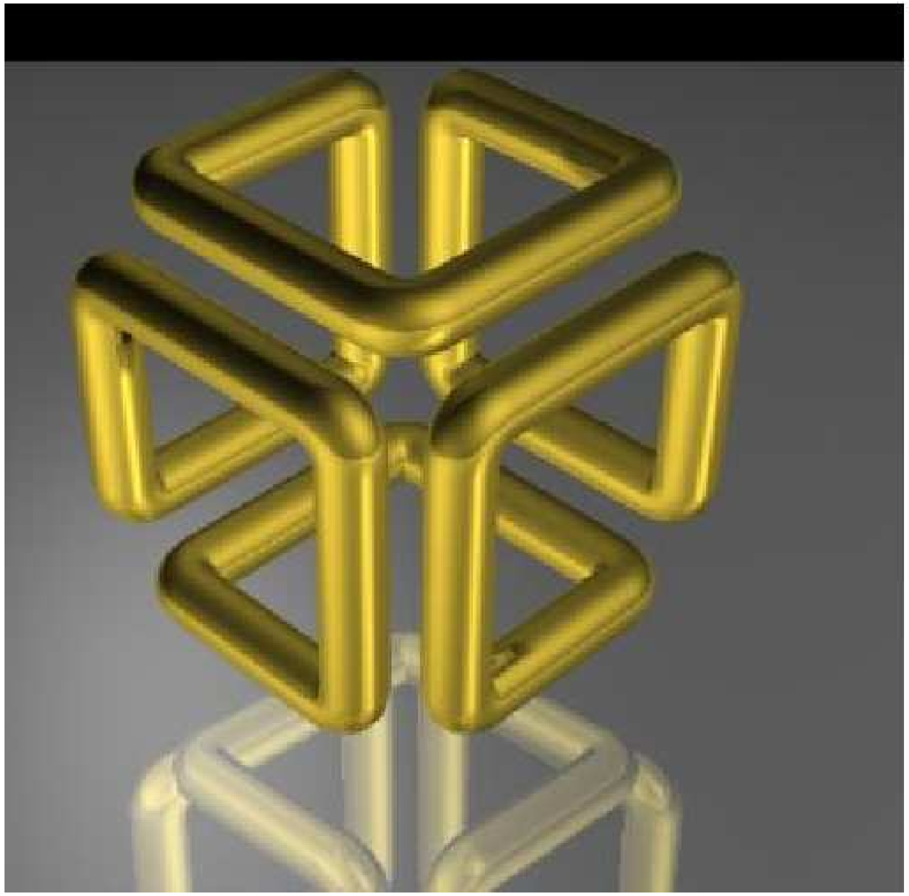
Sample Renderings (1)

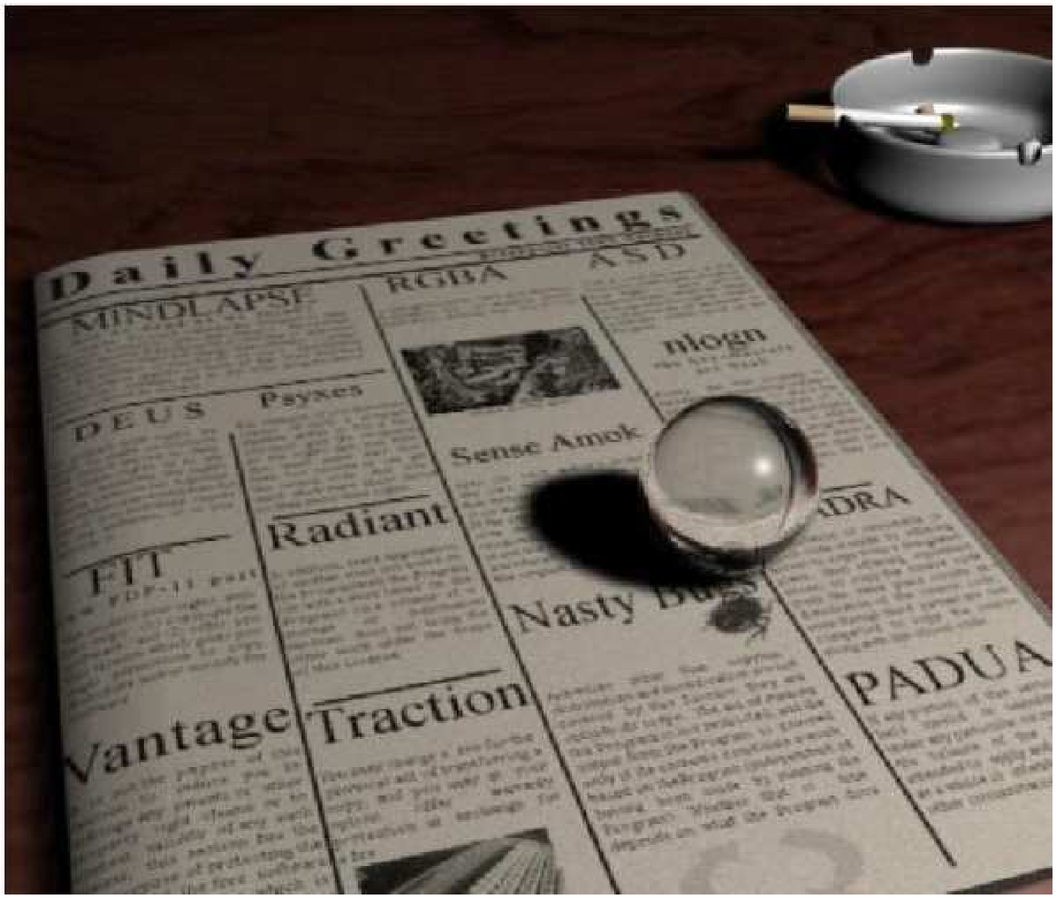


Sample Renderings (2)



Sample Renderings (3)





The End