Adaptive Scientific Visualization System for Desktop Computers and Mobile Devices

Konstantin Ryabinin,
e-mail: kostya.ryabinin@gmail.com

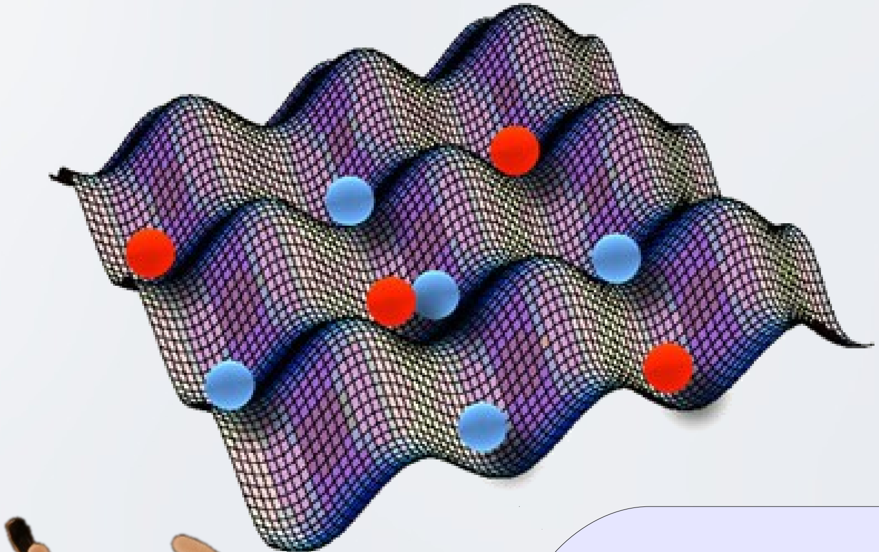
Svetlana Chuprina,
e-mail: chuprinas@inbox.ru

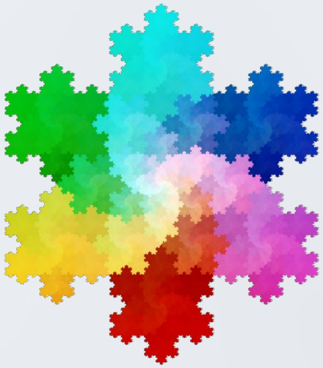
Perm State National Research University,
Bukireva Str. 15, 614990, Perm, Russia

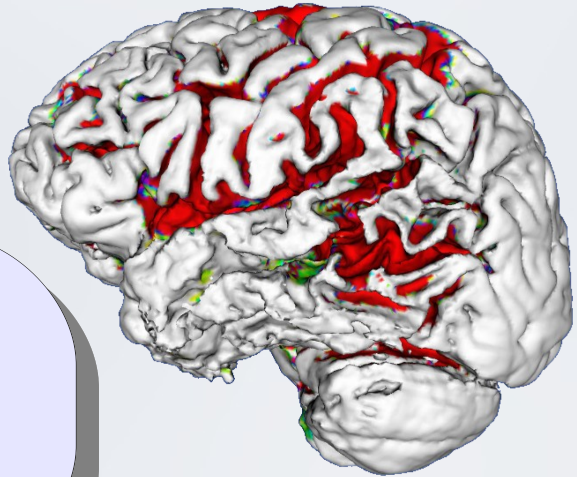
Barcelona, 2013

Scientific visualization

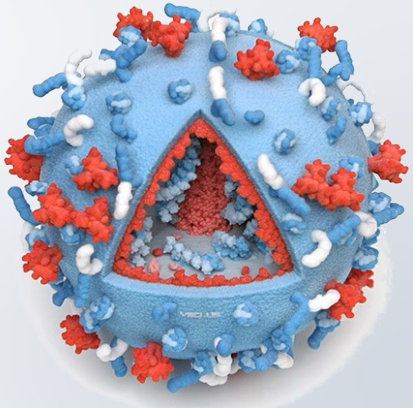
> **If I can't picture it,**
> **I can't understand it**
>
> *Albert Einstein*

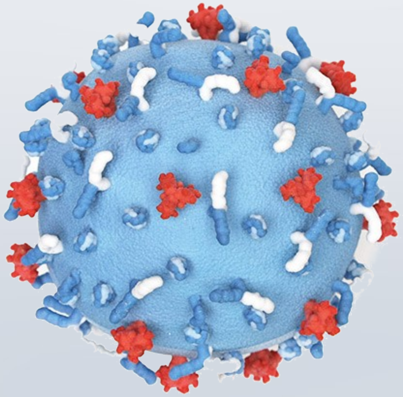


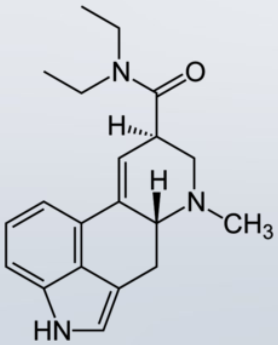


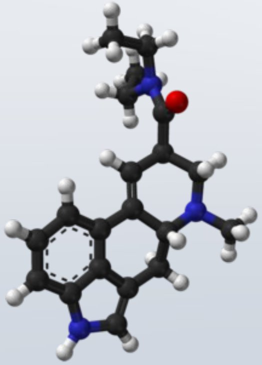


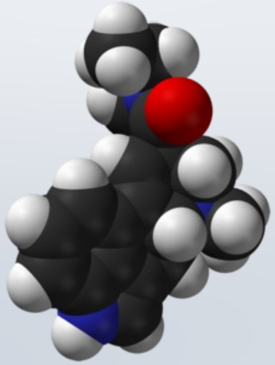
Scientific visualization –
essential part of
scientific research











Actual problems (1)

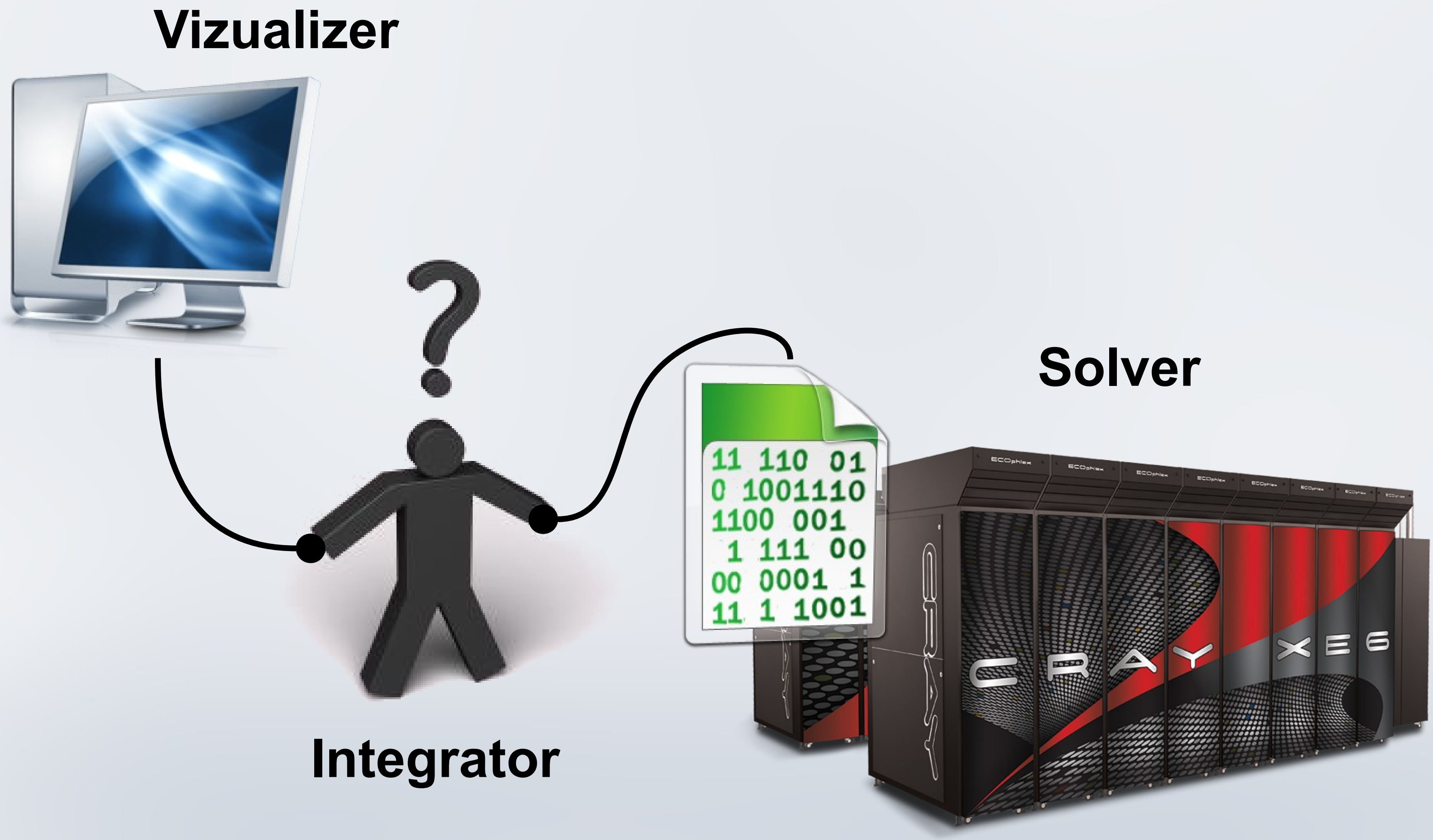
There are a lot of scientific visualization software

BUT

- There are no high-level integration means between solver and visualizer
- There are no adequate tools for multi-platform portability
- Implementation of distributed solvers and visualizers is inefficient (visualizers are not adaptive)

Actual problems (2)

- There are no high-level integration means between solver and visualizer



Actual problems (3)

- There are no adequate tools for multi-platform portability

Supercomputers

iOS

android

Mobile devices

Desktop computers

GNU / Linux

Mac OS

Microsoft Windows

Actual problems (4)

- **Implementation of distributed solvers and visualizers is inefficient**

Client-server visualization

There are 3 approaches to client-server visualization:

- **Visualization on the server-side (VNC)**
  - 🔴 **Low interactivity**
  - 🟢 **Low load on the client**
- **Visualization on the client-side**
  - 🟢 **High interactivity**
  - 🔴 **High load on the client**
- **Distributed visualization**
  - 🟢 **High interactivity**
  - 🟢 **Optimal load on the client and server**

Purpose (1)

Development of the client-server scientific visualization system

Main features:

- **Automatic integration with different solvers**
- **Adaptive distribution of rendering between client and server**
- **Multiplatform portability**

Purpose (2)

- Multiplatform portability
  - Server
    - Desktop computer
    - High-performance massively parallel supercomputer
  - Client
    - Desktop computer
    - Mobile device (smartphone / tablet)

Objectives

- Analysis of existing scientific visualization systems and tools
- Research of techniques to create multi-platform applications
- Developing multi-platform core of scientific visualization system (client and server)
- Developing tools to port graphical user interface across different platforms
- Reaching adaptivity of system to performance of the client and speed of the connection
- Testing of the visualization system implementation for a lot of different real scientific problems

About applications

Applications that have

- solver and visualizer at once:
    - MathCad, MatLab, Mathematica, Maxima, …
    - PocketCAS, GraphCalc, ...
- visualizer only:
    - TecPlot, Origin, EasyPlot, IRIS Explorer, Surfer, Grapher, AMLab, ParaView …
    - KiwiViewer

Drawbacks:

- Only a few mobile solutions
- Only a few multi-platform solutions
- Only a few solutions allow automatic integration with solver running on high-performance computer

About libraries

- Visualization Toolkit:
  - pVTK (for the massively parallel server)
  - VTK (for the desktop client)
  - VES (for the mobile client)

About graphics rendering engines

- Unreal Engine, SIO2, Unity3D
- OGRE, Irrlicht, Oolong, Cocos 2D / 3D, libGDX

Drawbacks:

- Orientation not to scientific visualization,
  but only to games
- No support of special scientific
  rendering techniques
  (volume rendering, slice-based rendering, etc.)

About GUI frameworks

- Libraries and frameworks:
  - Qt, GTK, Tk, Awt / Swing
  - Clutter, PhoneGap
- HTML5 / CSS:
  - MoSync
- Flash

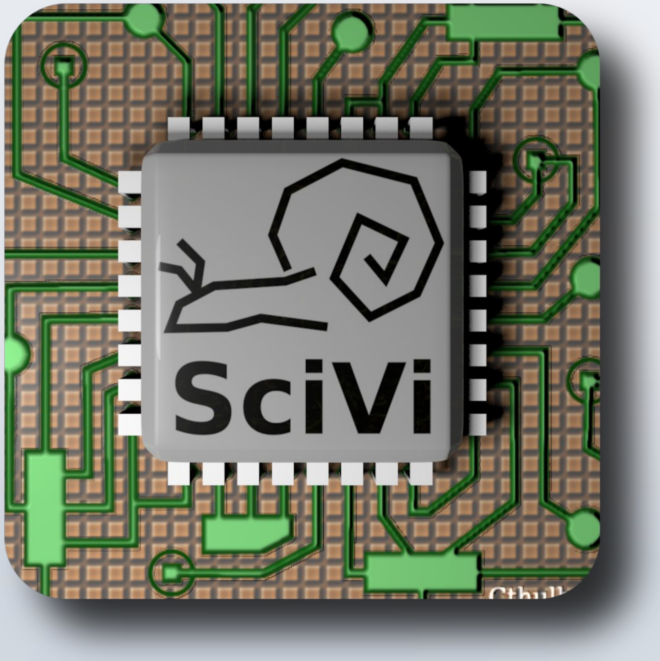
Drawbacks:

- Insufficient efficiency
- Stability problems on mobile platforms
- Double-design problem: GUI should be designed twice (for desktop computer and for mobile device)

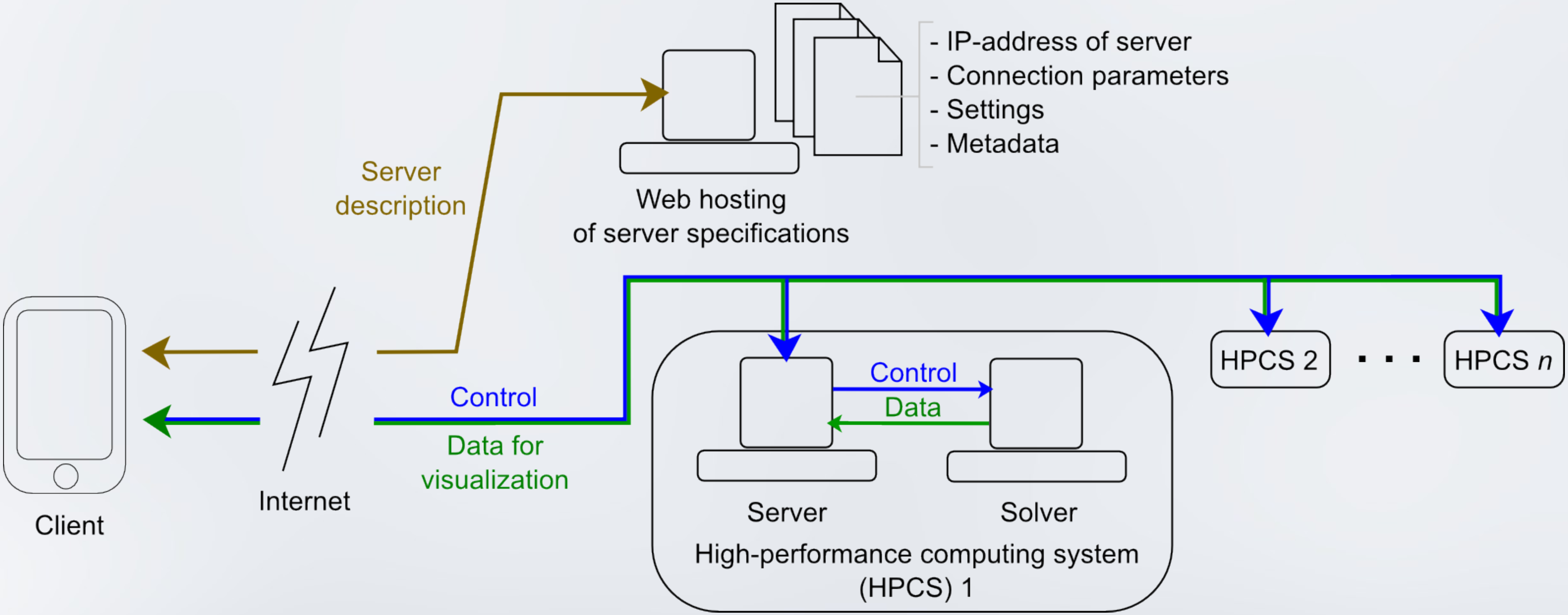
Proposed solution

Scientific visualization system called *SciVi*

- Client-server architecture
- Multi-platform portability
- Adaptive distribution of rendering between client and server
- Automatic integration with solver
- Special rendering techniques
- Dublin Core as metadata standard
- High-level tools for describing solvers



SciVi: architecture



SciVi: multi-platform portability (1)

- Inplementation in C++
- Usage of OpenGL(ES) as a rendering standard
- Usage of *NFoundation* и *NGraphics* libraries (developed in Russian IT-company Nulana LTD located in Perm) as an abstraction layer between OS and SciVi
- Solving of GUI framework problem:
  - Development of our own GUI framework based on OpenGL(ES)

SciVi: multi-platform portability (2)

- **Solving double-design problem:**
  - **Development of GUI builder adaptive to concrete platform based on high-level declaration in XML**
- **Target platforms:**
  - **Windows, GNU / Linux, Mac OS X**
  - **iOS, Android**

SciVi: distributed rendering

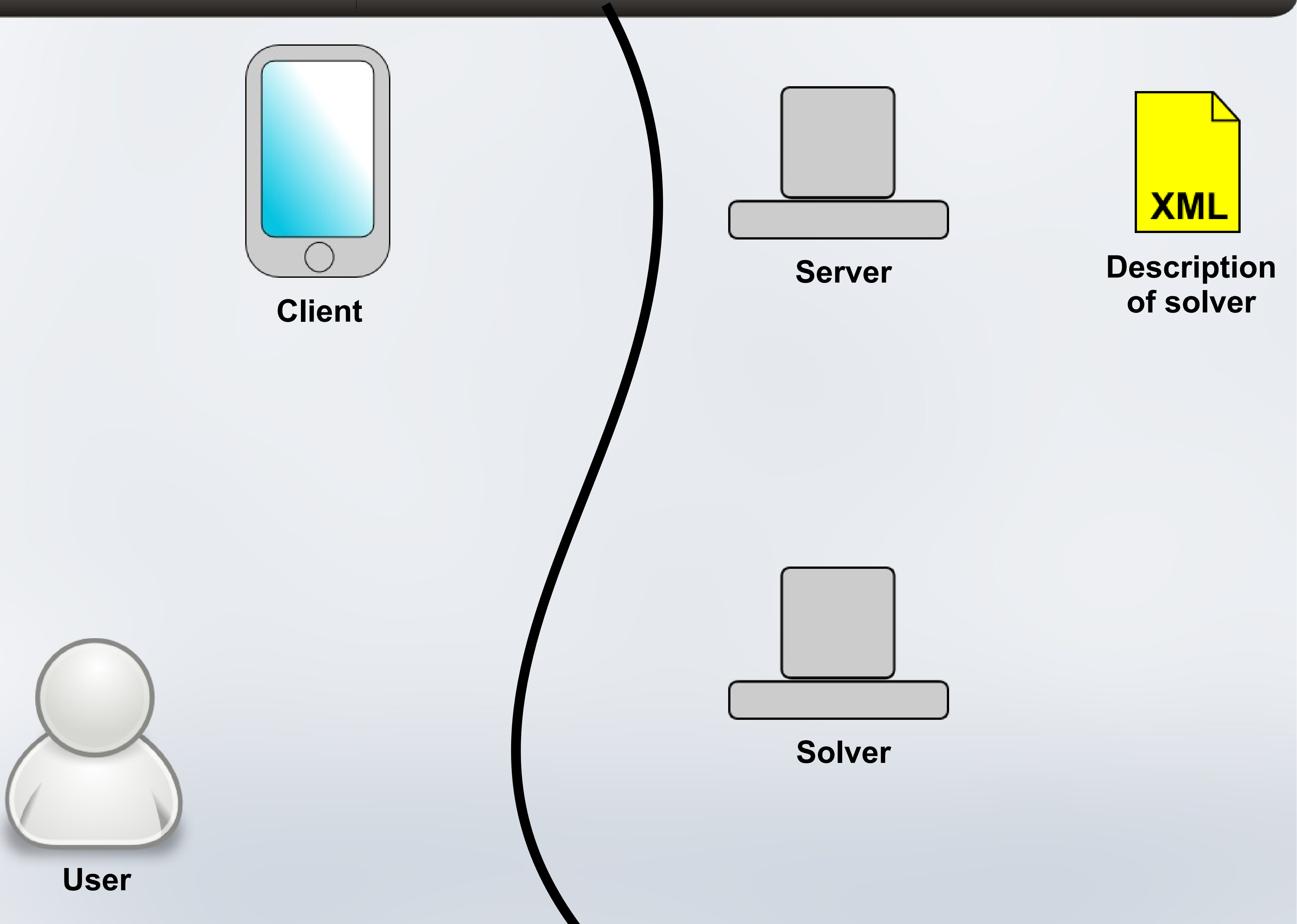
Server-side:

- Planning distribution of rendering based on heuristics
- Rendering of static parts of scene (textures, etc.)
- Rendering using pVTK
- Preprocessing of data before sending to the client
- Adaptivity to performance of the client and speed of the connection through simplification of data

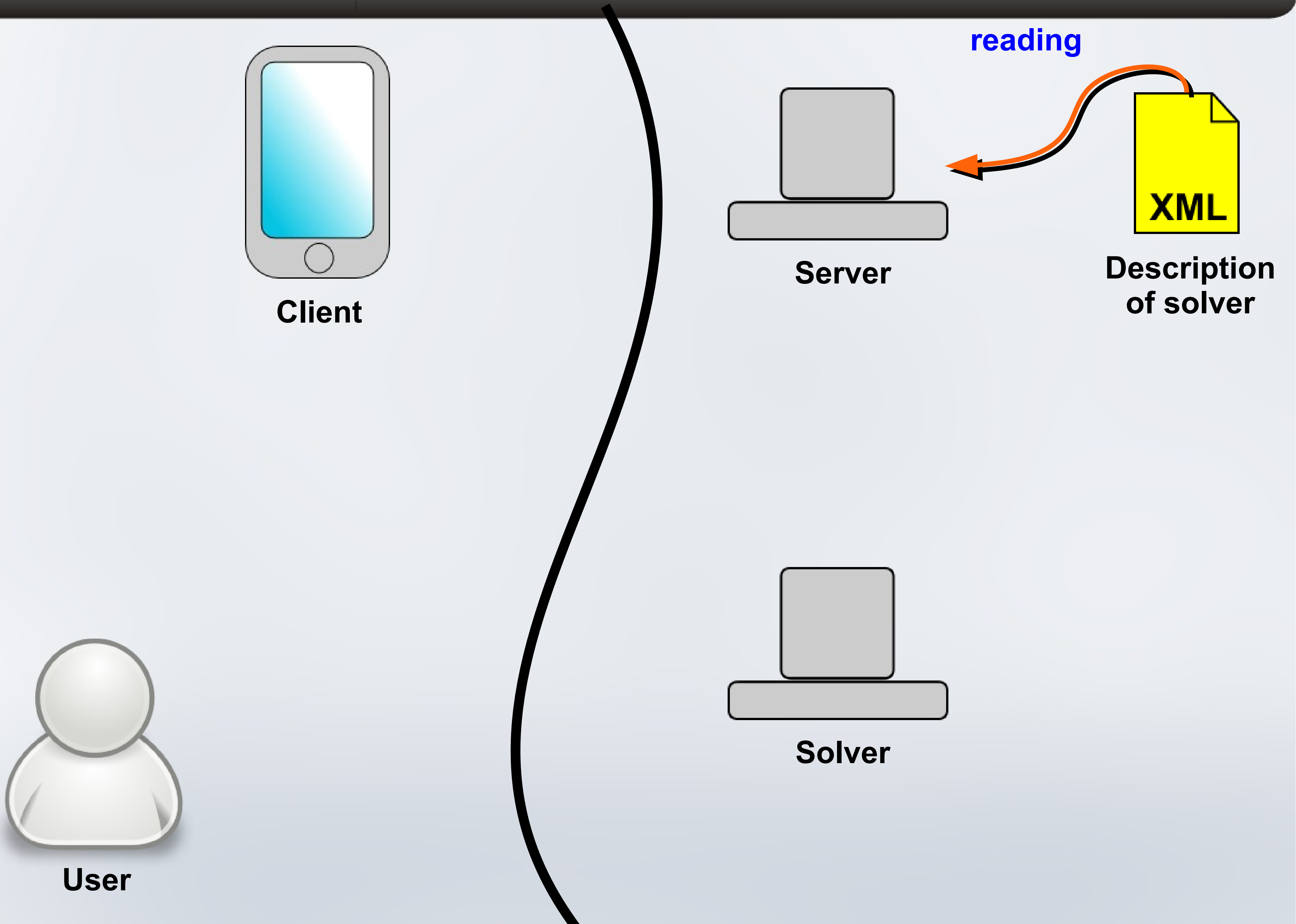
Client-side:

- Renedring of final image using our own algorithms and VTK / VES
- High interactivity

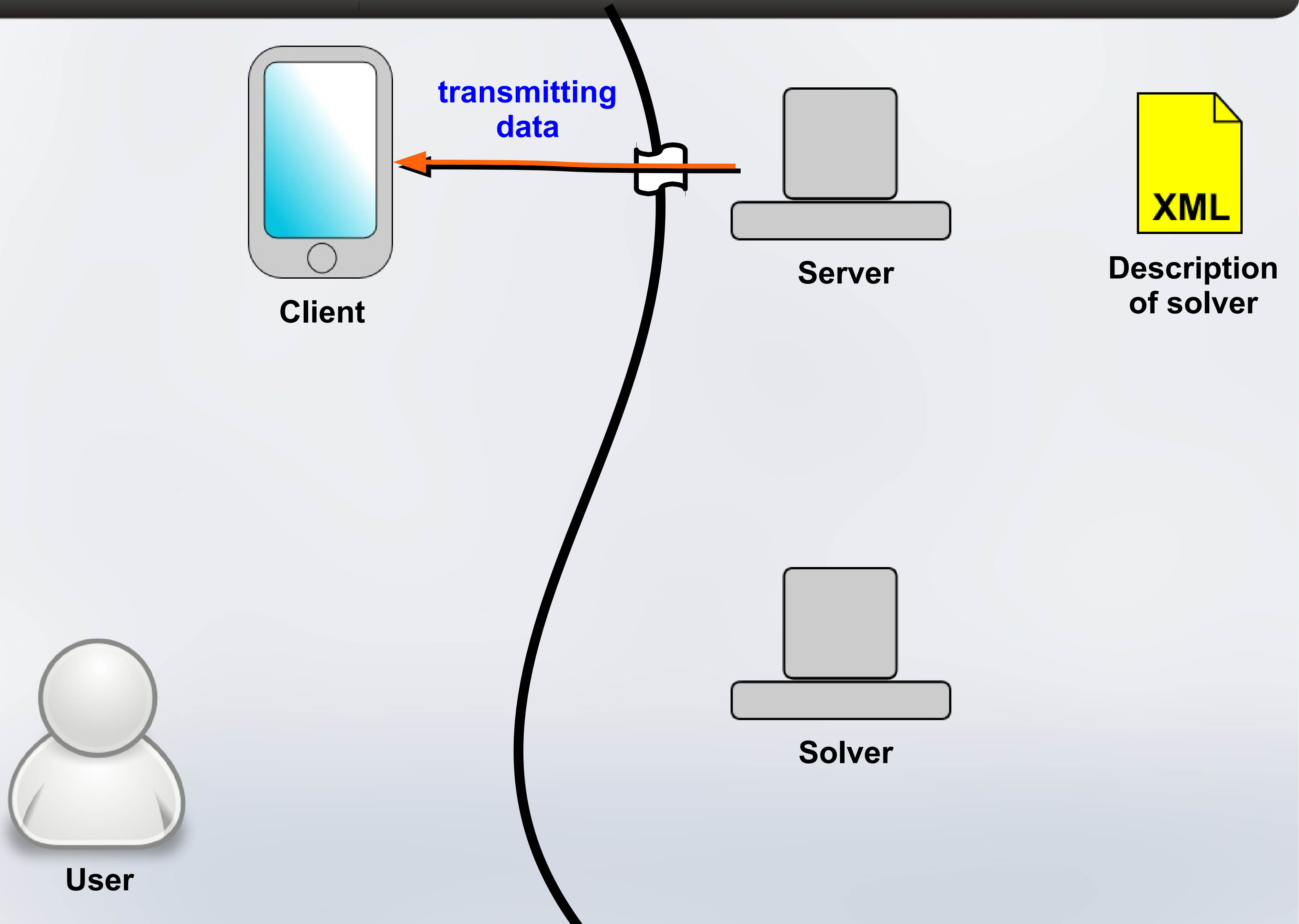
SciVi: intergation with solver (1)



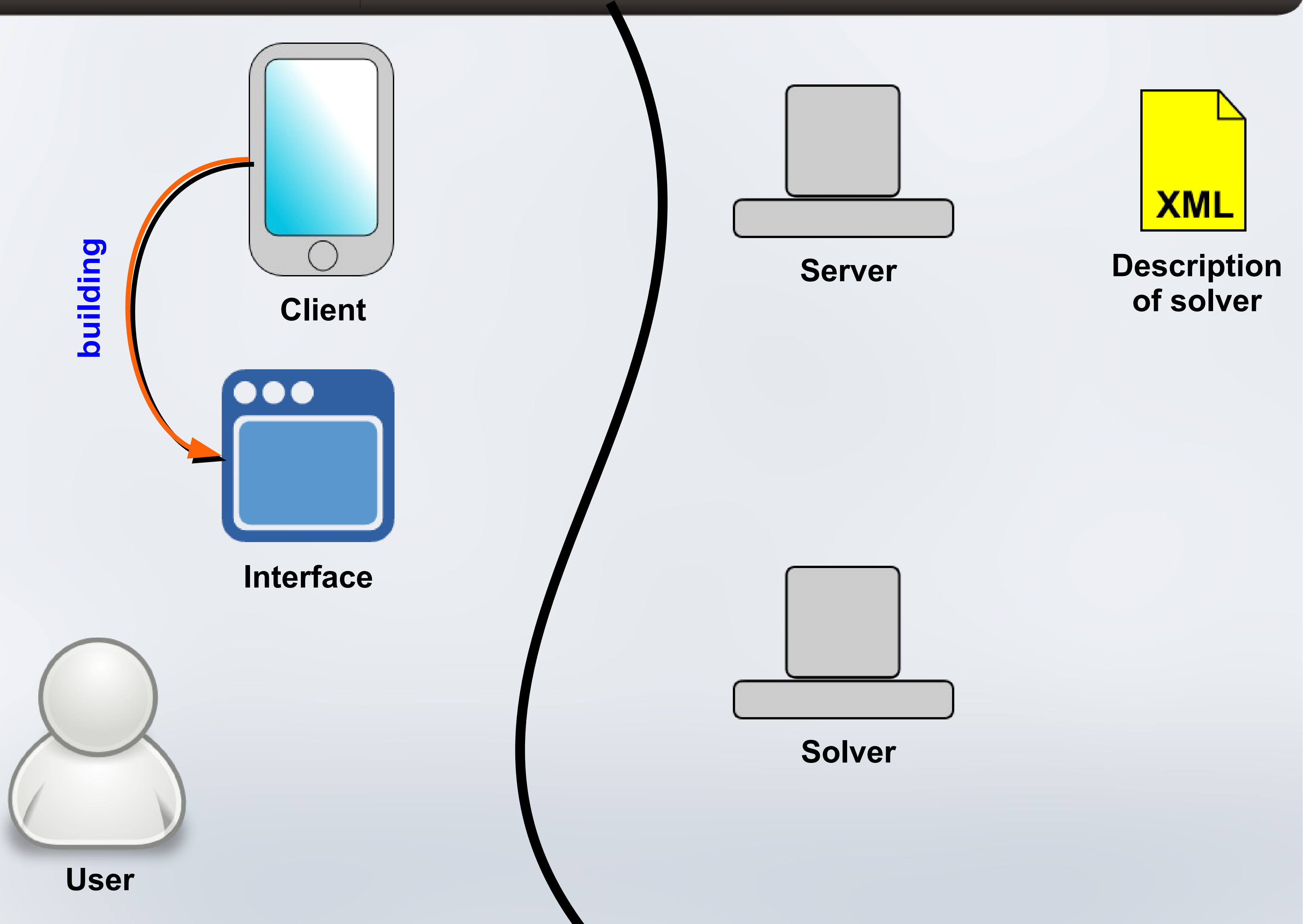
SciVi: intergation with solver (2)



SciVi: intergation with solver (3)



SciVi: intergation with solver (4)



building

Client

Interface

User

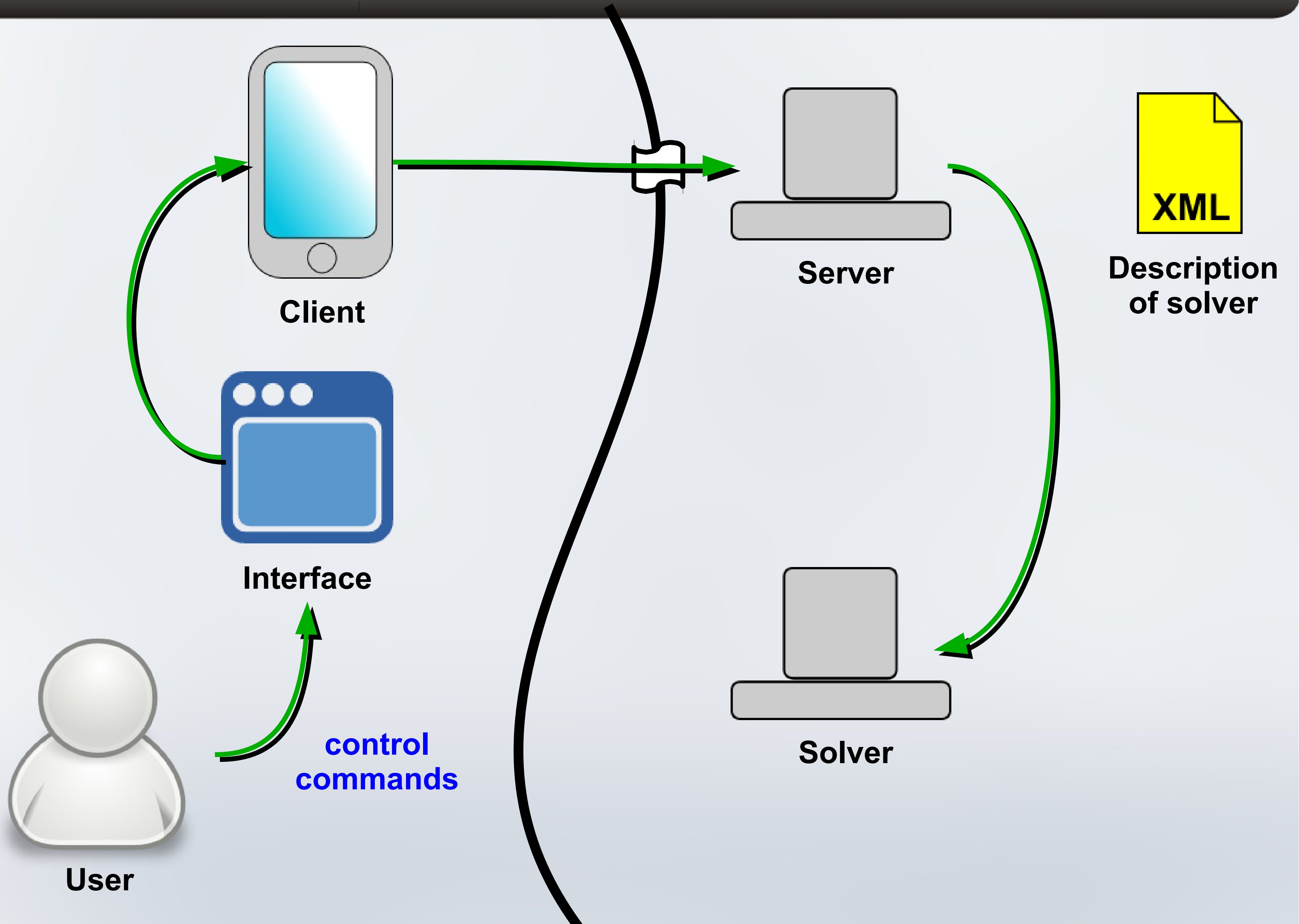
Server

XML

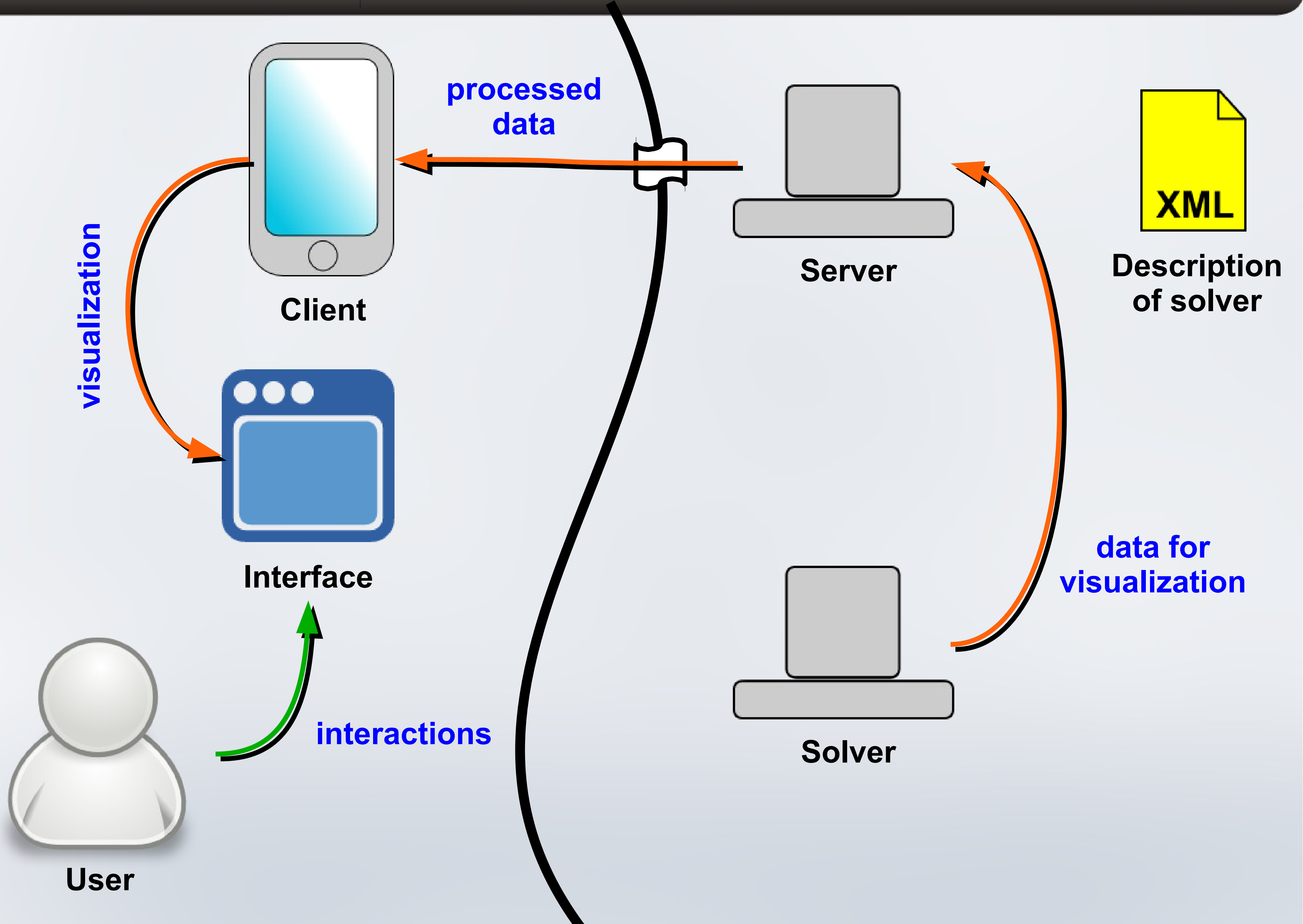
Description of solver

Solver

SciVi: intergation with solver (5)



SciVi: intergation with solver (6)

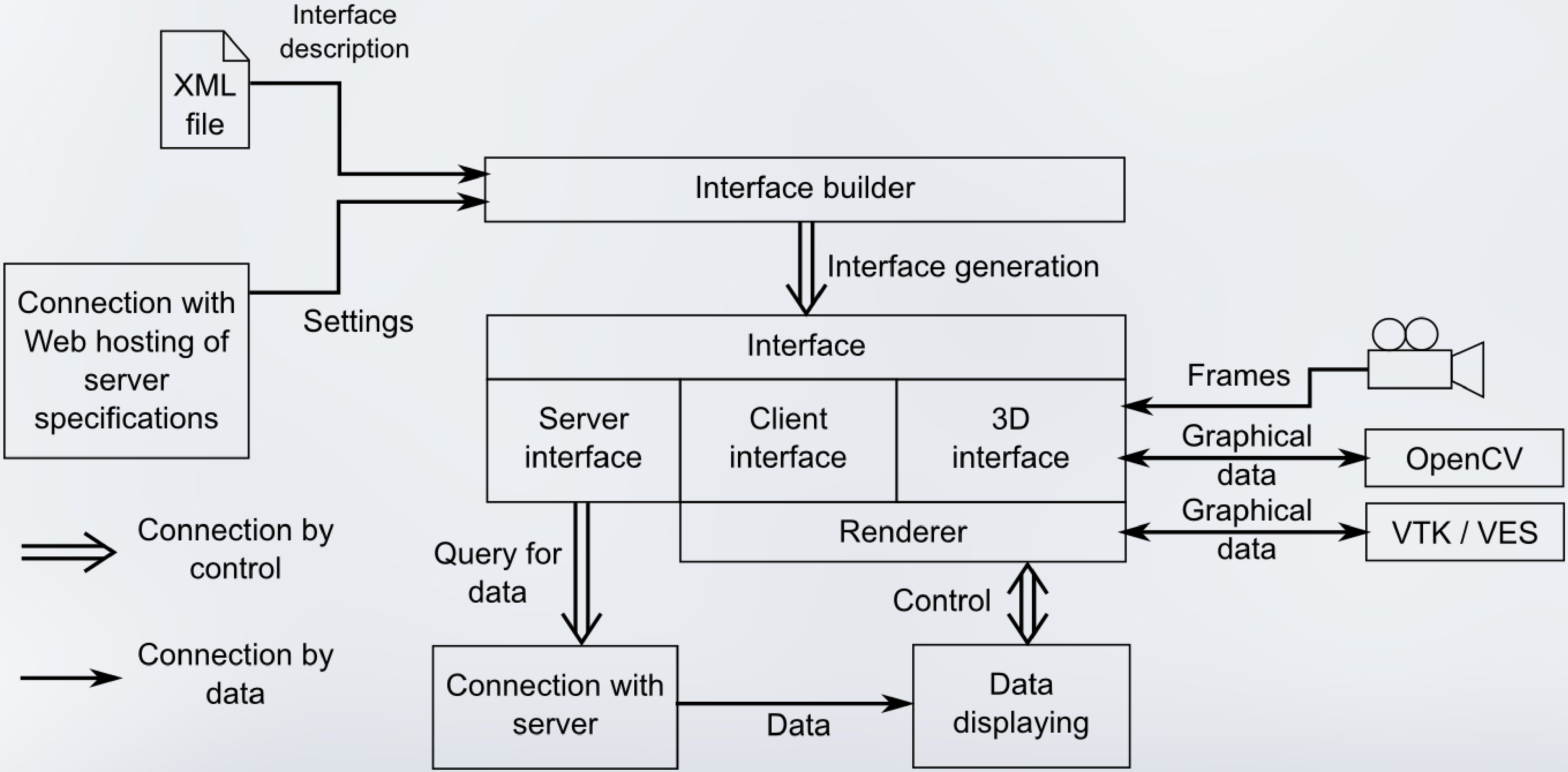


SciVi: graphics rendering

Current version:

- Rendering of shaded and textured surfaces and 3D-models
  - Custom GLSL/ESSL shaders
  - 3D-moldes in PLY, 3DS and internal binary format
  - Textures in PNG, JPG, BMP, DIB and TGA
  - Basic primitives (spheres, cubes, billboards, etc.)
- Multiscale rendering (microscope metaphor)
- Animation
  - Affine transform
  - Uniform-variables in shaders
- XML-description of scene with links to resources (textures, models, shaders source code, etc.) obtained from server

SciVi: architecture of client

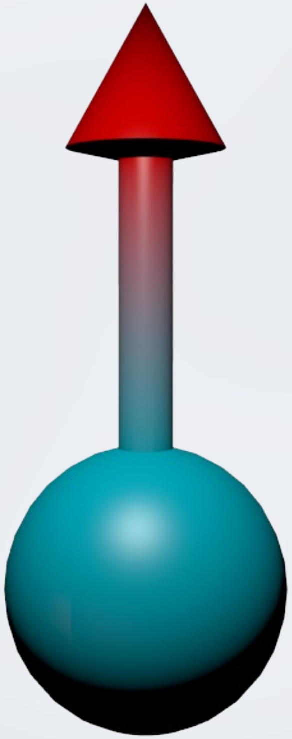


SciVi: support of Dublin Core

- Necessary resources are annotated according to Dublin Core standard to make them searchable and reusable
- Example of metadata for electron's spin model:

spin.xml:

```
<meta name="DC.Title"
      content="Spin of electron"/>
<meta name="DC.Creator"
      content="Konstantin Ryabinin"/>
<meta name="DC.Date"
      content="04.05.2013"/>
<meta name="DC.Source"
      content="PLY"/>
<meta name="DC.Format"
      content="N3D"/>
<meta name="DC.Description"
      content="3D-Model of electron's spin"/>
<meta name="DC.RightsHolder"
      content="Perm State National Research University"/>
```



spin.n3d

Example: modelling of electron's spins in magnetic filed (1)

- **Solver has been developed in Perm State National Research University**
- **Solver runs on supercomputer and produces output looking like**
- **Example of solver's output**

$$
\begin{array}{lll}
t_1 & & \\
x_1^1 & y_1^1 & z_1^1 \\
x_2^1 & y_2^1 & z_2^1 \\
\ldots & & \\
x_n^1 & y_n^1 & z_n^1 \\
t_2 & & \\
x_1^2 & y_1^2 & z_1^2 \\
x_2^2 & y_2^2 & z_2^2 \\
\ldots & & \\
x_n^2 & y_n^2 & z_n^2 \\
\ldots & &
\end{array}
$$

t_j – modelling time

$x_i^j \, y_i^j \, z_i^j$ – coordinates of spin i in t_j moment of time,

$i=\overline{1,n}, \; j=\overline{1,m}$

Example: modelling of electron's spins in magnetic filed (2)

- **Example of concrete output data**

```
0
0.440798910779106 0.193456073216762 -0.876510734669864
0.0480228711779573 -0.284014985805231 -0.957616463769227
0.36786492397633 0.792218280706476 -0.486893821507692
-0.114873074685615 0.0755407821401694 -0.990503794513643
. . .
0.35
0.501727936749538 -0.00133378223902795 -0.865024449660234
-0.00346408984151143 -0.272264254539676 -0.962216283265631
0.628307647042774 0.620768861805486 -0.468908862019364
-0.0396858270658191 0.11017684776047 -0.993119377188698
. . .
```

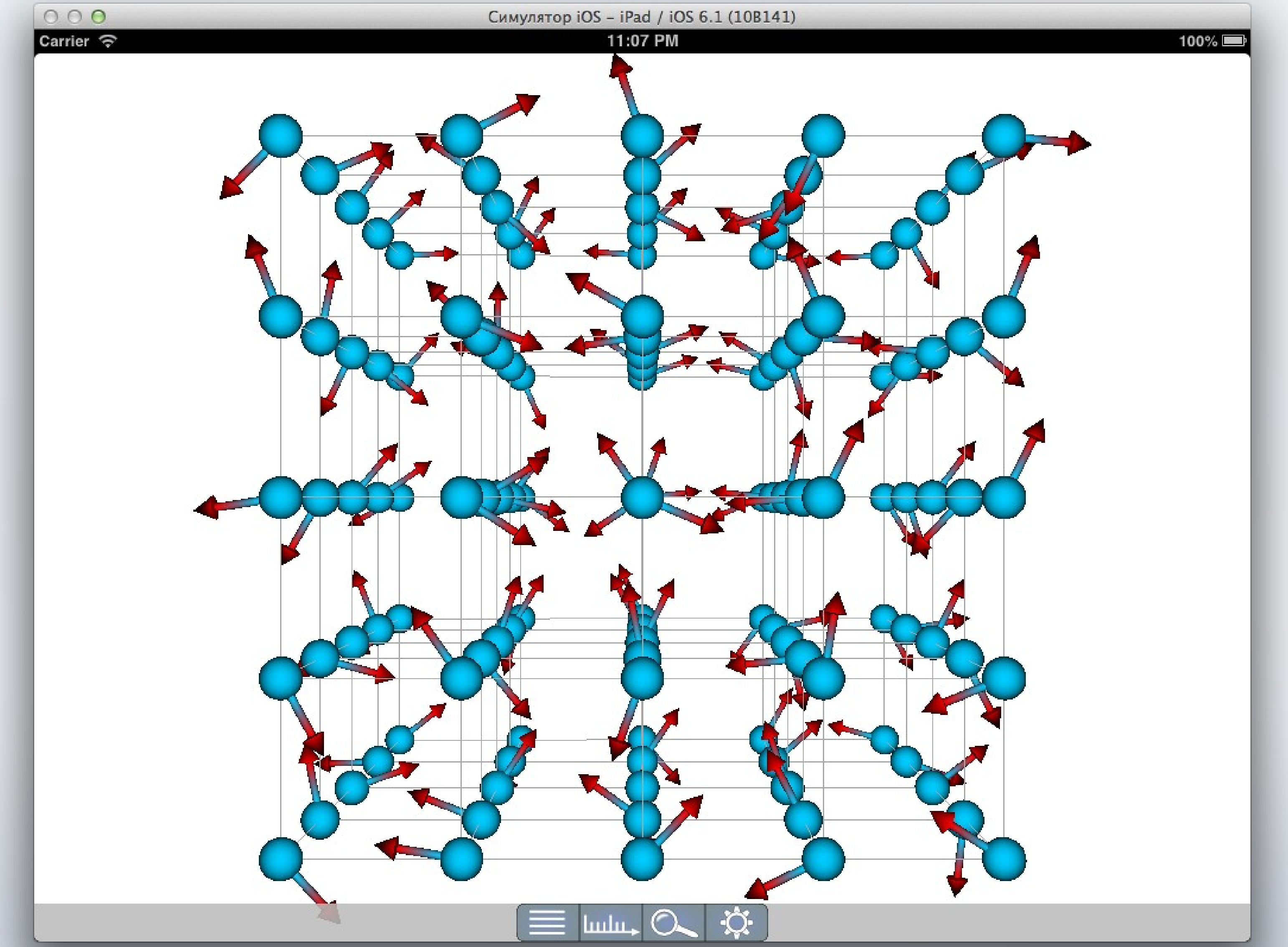
Example: modelling of electron's spins in magnetic filed (3)

- **Server transforms the output data to the XML-description of scene, acceptable for SciVi**

```
<model id="spin_0">
  <data model="http://dl.dropbox.com/u/71028668/scivi/spins/spin.n3d"
        shader="blinn_vc"/>
  <position x="-0.750000" y="-0.750000" z="-0.750000"
            scaleX="0.045" scaleY="0.045" scaleZ="0.045"
            dirX="0.440799" dirY="0.193456" dirZ="-0.876511"/>
  <animation>
    <timestamp id="1">
      <position dirX="0.501728" dirY="-0.001334" dirZ="-0.865024"/>
    </timestamp>
    <timestamp id="2">
      <position dirX="0.492641" dirY="-0.206796" dirZ="-0.845305"/>
    </timestamp>
    <timestamp id="3">
      <position dirX="0.428445" dirY="-0.370627" dirZ="-0.824057"/>
    </timestamp>
    . . .
  </animation>
</model>
. . .
```

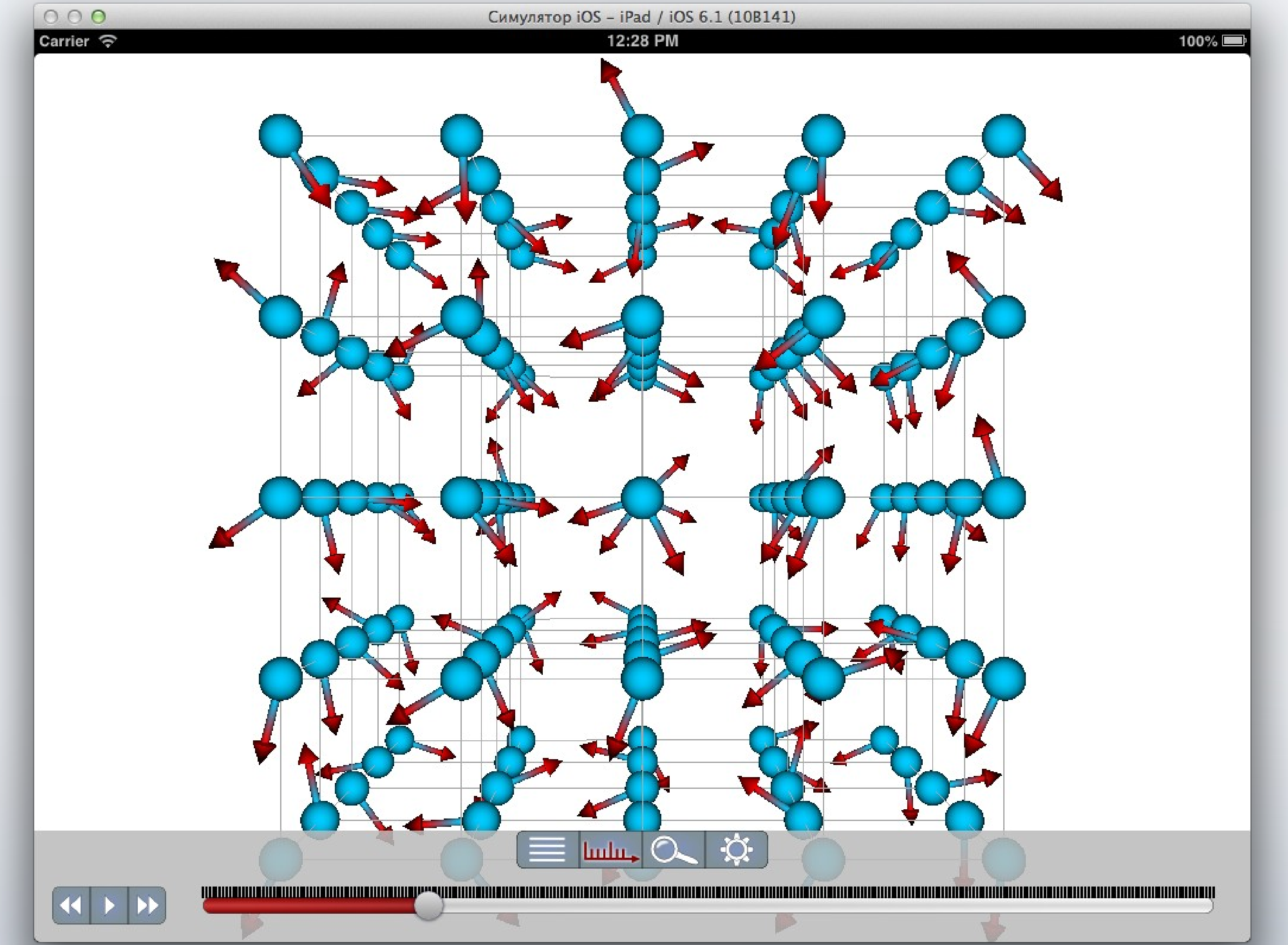
Example: modelling of electron's spins in magnetic filed (4)

- **The result of client-side visualization**



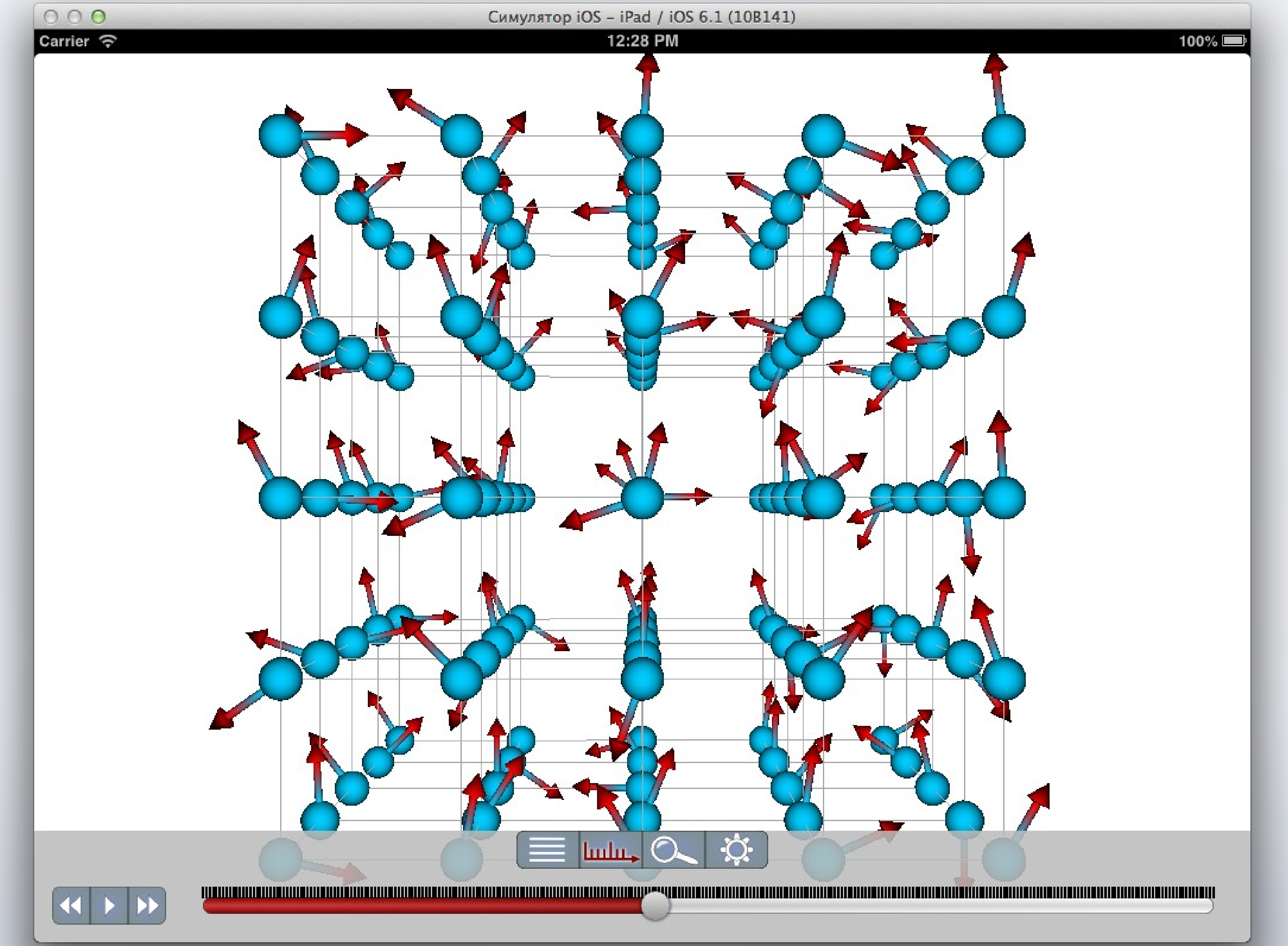
Example: modelling of electron's spins in magnetic filed (4)

- **The result of client-side visualization**



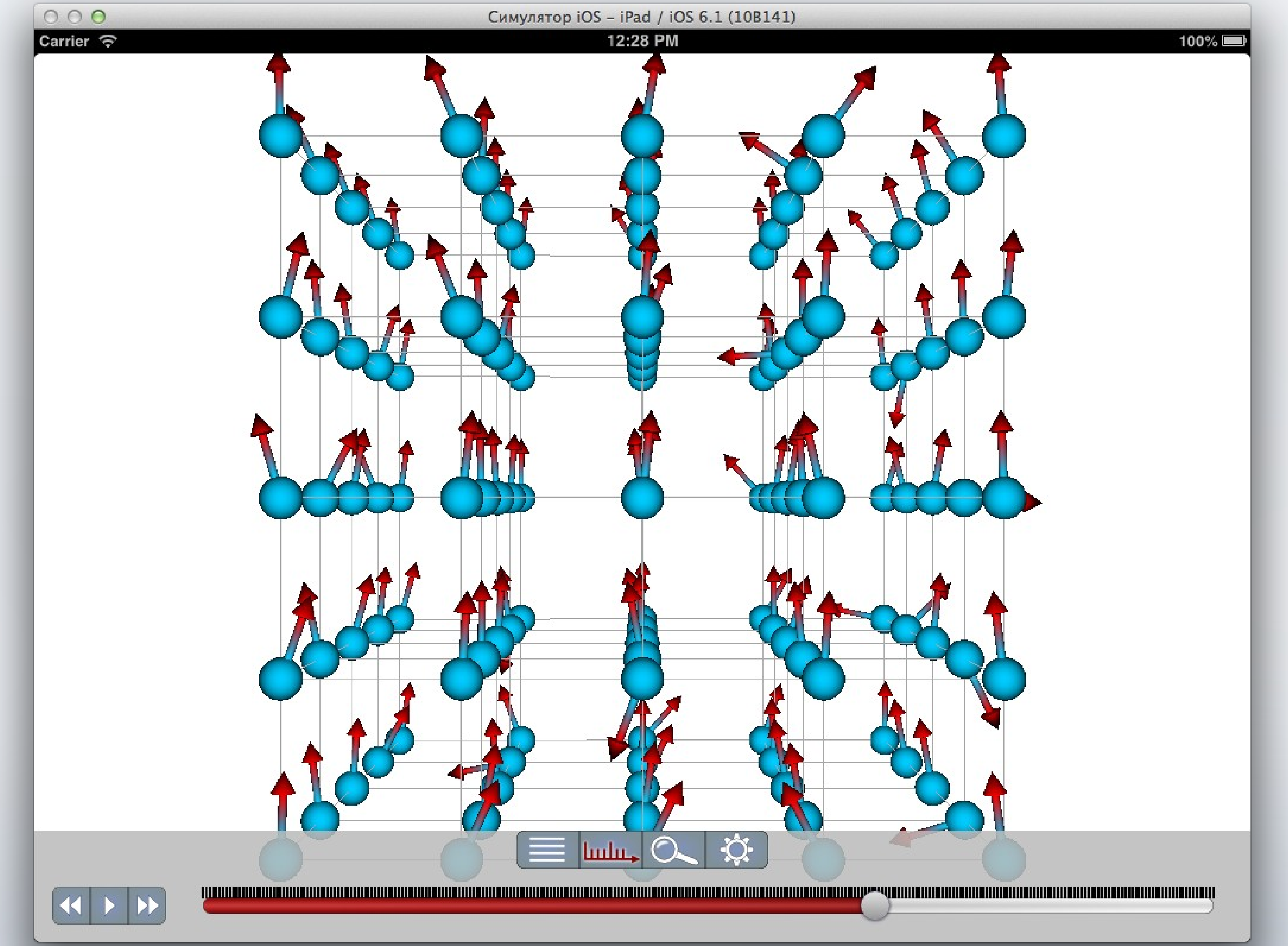
Example: modelling of electron's spins in magnetic filed (4)

- **The result of client-side visualization**



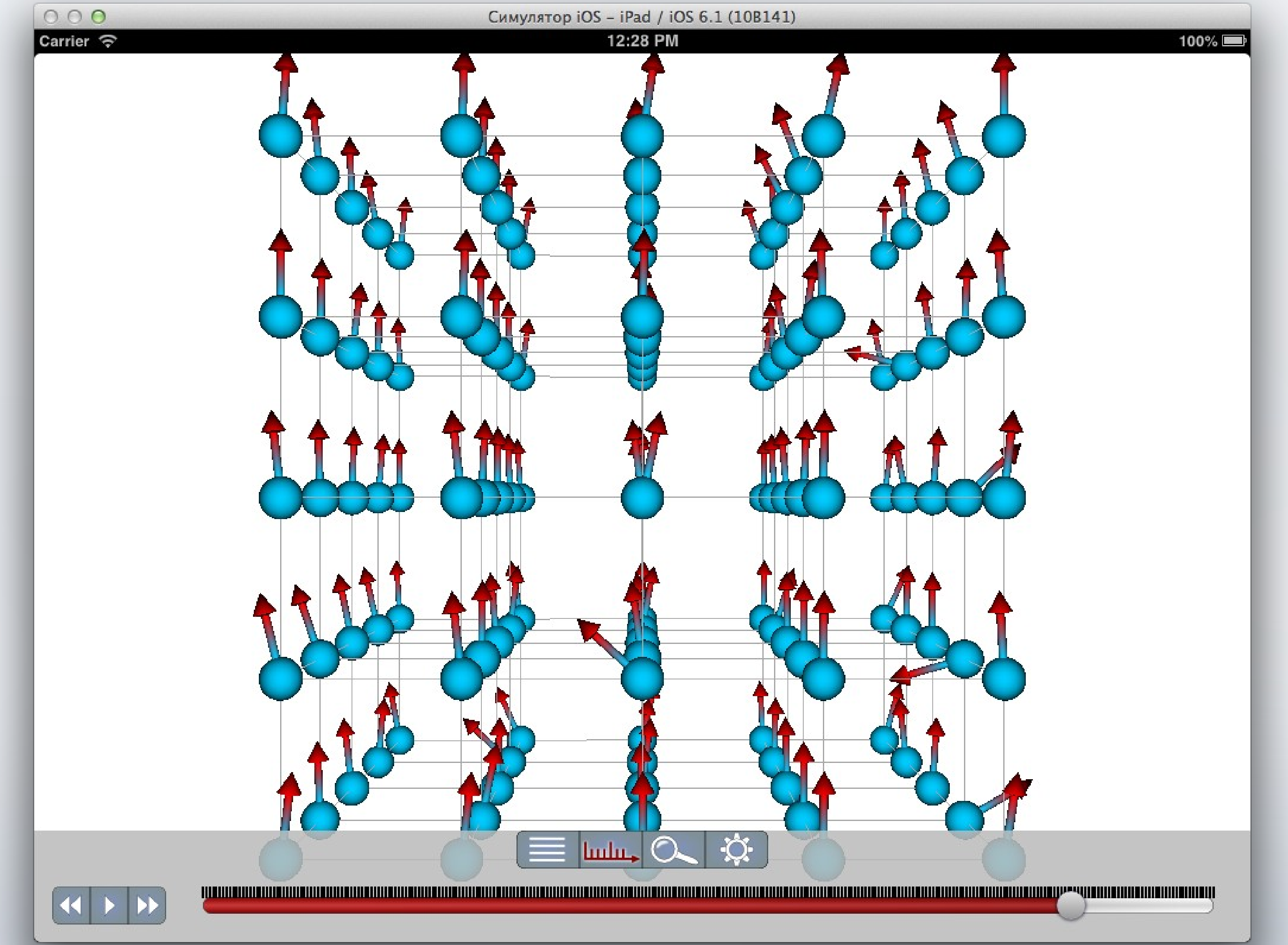
Example: modelling of electron's spins in magnetic filed (4)

- **The result of client-side visualization**



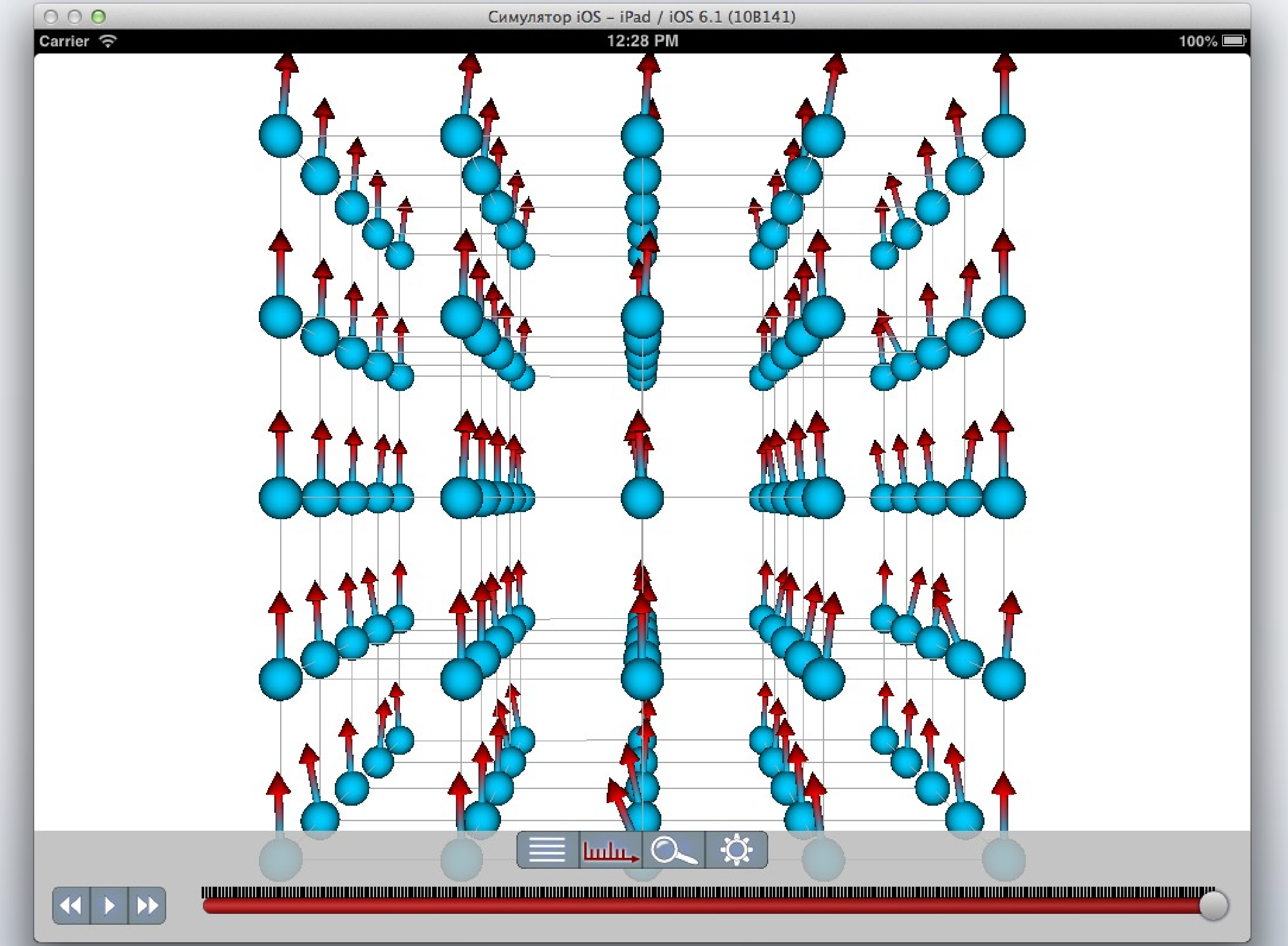
Example: modelling of electron's spins in magnetic filed (4)

- **The result of client-side visualization**



Example: modelling of electron's spins in magnetic filed (4)

- **The result of client-side visualization**



Conclusions: tasks completed

- Working prototype of client
  - Subsystem that automatically builds graphical user interface
  - Subsystem that renders 3D-scene in multiscale mode
  - Subsystem that communicates with web-hosting of server settings
  - Multiplatform portability:
    support of Windows, GNU / Linux, Mac OS X, iOS and Android
- Demo-prototype of server

Conclusions: future plans

- Improvement of client
  - Integration with technologie of augmented reality
  - Integration with VTK / VES
  - Implementation of multidimensional data rendering
  - Implementation as a library
- Implementation of server working prototype
  - Development of high-level tools for integration with solvers (based on ontologies)
  - Adaptivity to solver, client and speed of connection
  - Distribution of rendering
- More tests on different real scientific problems

Thank you for your attention!

Konstantin Ryabinin,
e-mail: kostya.ryabinin@gmail.com

Svetlana Chuprina,
e-mail: chuprinas@inbox.ru

Perm State National Research University,
Bukireva Str. 15, 614990, Perm, Russia

Barcelona, 2013