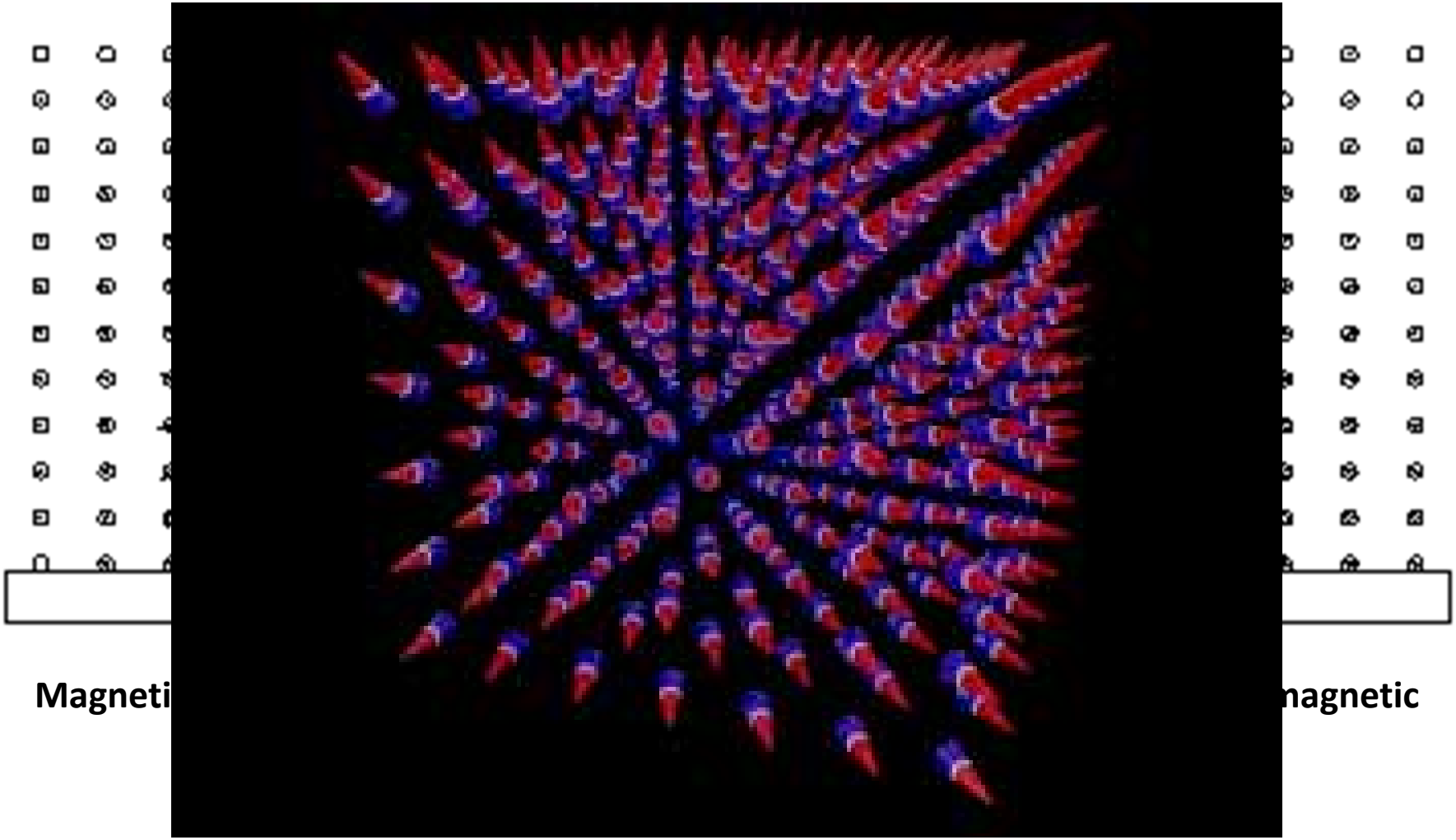
Prototyping of visualization styles of 3D scalar fields using POV-Ray rendering engine

J. Opiła

AGH University of Science and Technology,
Department of Applied Computer Science,
Faculty of Management, Cracow, Poland
jmo@agh.edu.pl

Long, long ago in a distant galaxy….



Magneti **nagnetic**

Brief introduction

Four visualization styles of 3D static scalar field employing POVRay ray-tracing engine are discussed:

- equipotential surface method using direct implementation of *isosurface*{} – POVRay built in object,
- cellular trilinear interpolation approach,
- application of texture or layered textures,
- pseudo-particles design.

Styles presented have been tested for hybrid visualizations and compared concerning computing time, informativeness and general appearance.

SDL Language

It is also shown that Scene Description Language (SDL), domain specific language implemented in POV-Ray is flexible enough to use it as a tool for fast prototyping of novel visualization techniques.

Visualizations discussed in the paper were computed using selected components of API of ScPovPlot3D, i.e. „templates” written in the SDL language:

www.scpovplot3d.sf.net

Virtual laboratory

In order to inspect phenomena one needs to conduct some experimental research.

However, all “experiments” to be done requires computer only, thus name “numerical experiments“.

Because numerical experiments were performed in “virtual laboratory”, next is explained how such “laboratory” has been designed.

Virtual laboratory, decisions

1. Choice of rendering engine,
    a. What ScPovPlot3D is and why has been used,
2. Choice of exemplary scalar field,
3. Field approximation, Lennard-Jones potential,
5. Computer.

1.Choice of rendering engine

BioBlender,

Blender,

Diderot,

gnuplot,

ParaView,

RasMol,

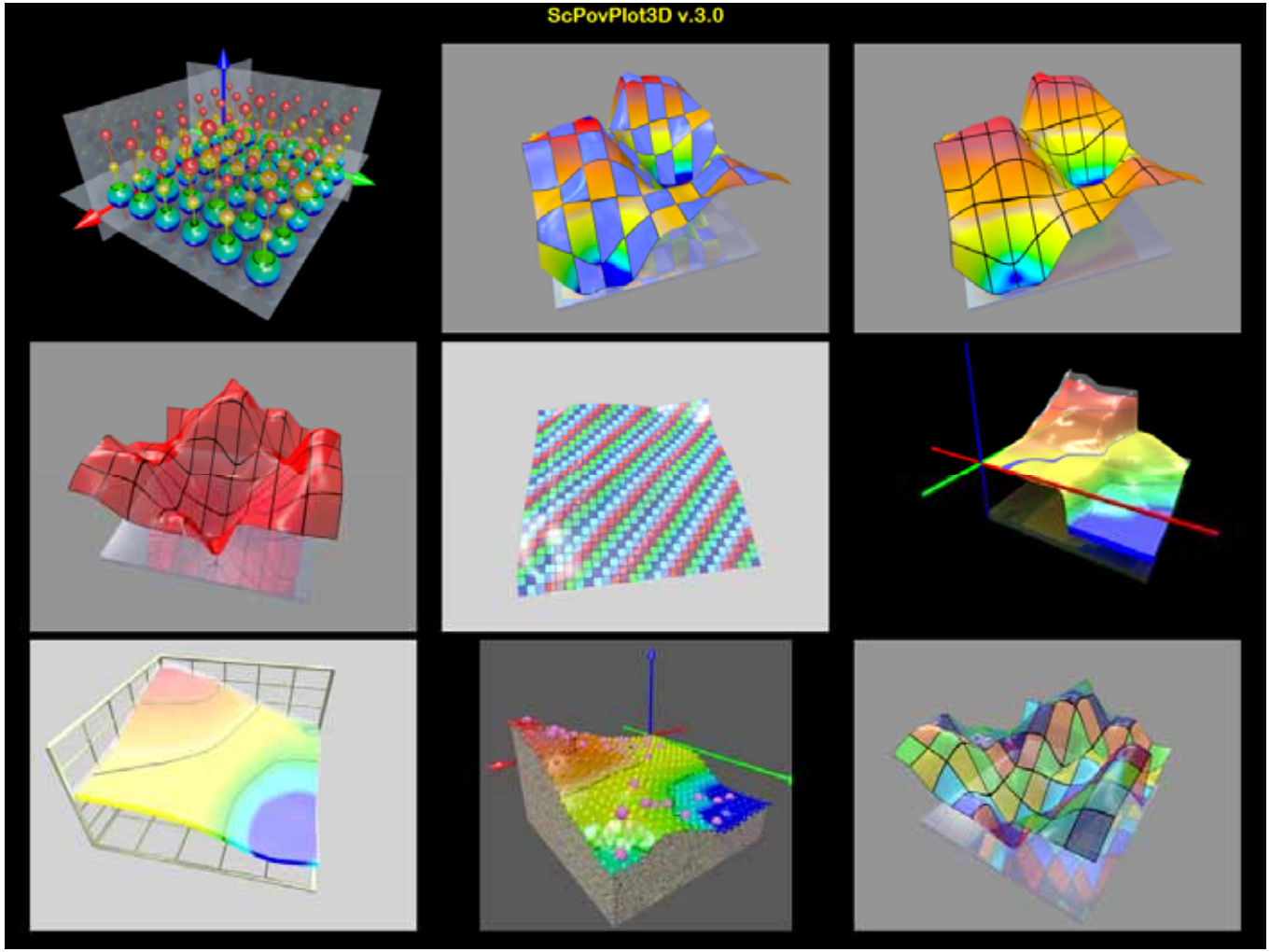
Shadie,

VTK/C++, VTK-Python

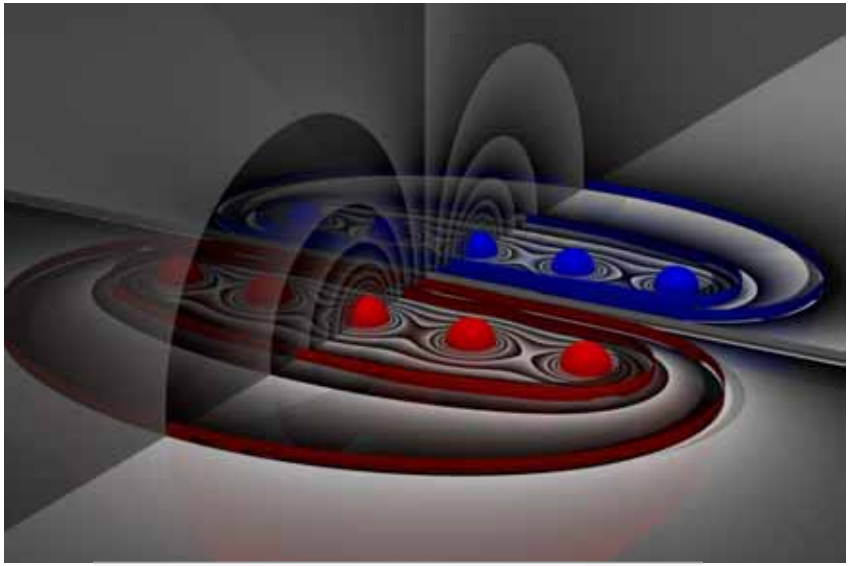
POV-Ray.

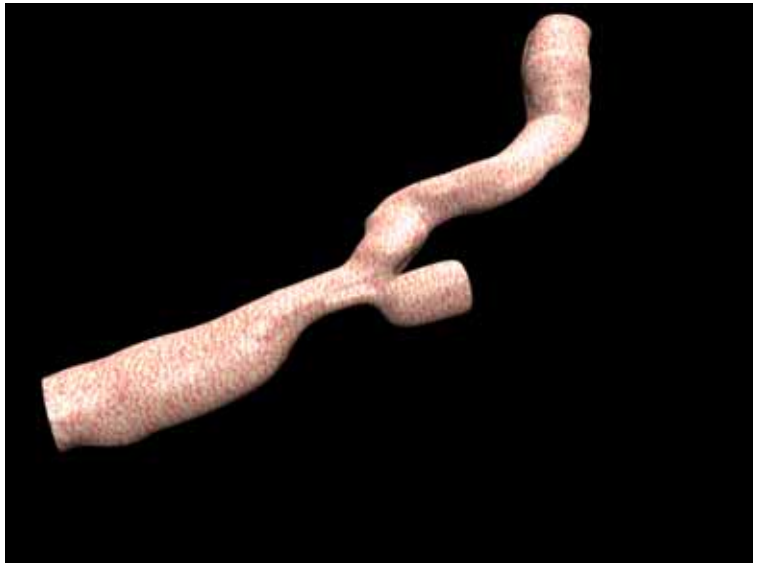
1a. POVRay (source – Hall Of Fame)

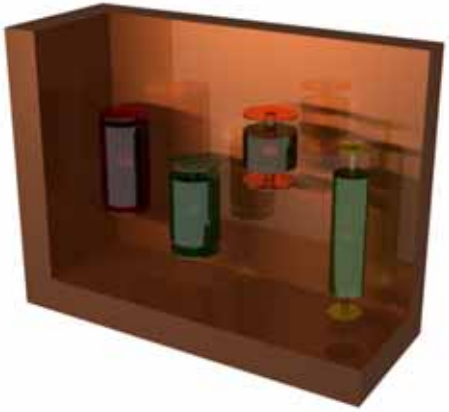
1b. ScPovPlot3D, www.scpovplot3d.sf.net



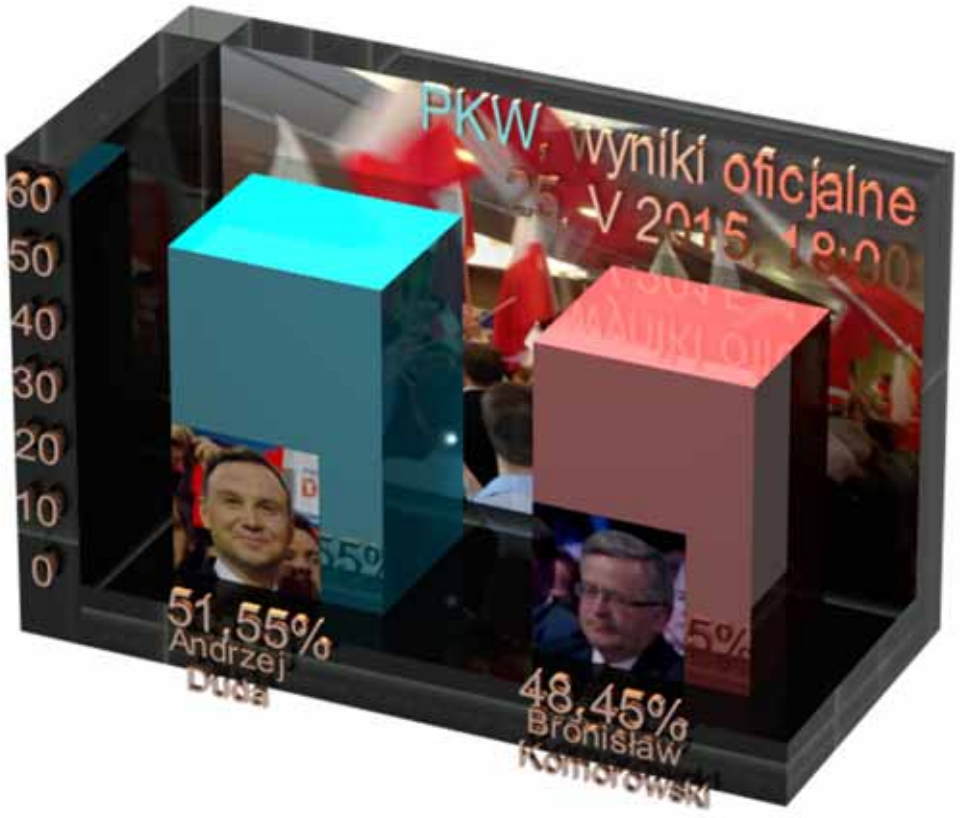
1c. many domain visualizations …







1d. not only scientific….



1e. POVRay IDE with L-J code snippet

POV-Ray - C:\Users\mirfak\Documents\POV-Ray\Sceny\dev02\examples\Plot3Dv3\Potential.inc [Idle]

File Edit Search Text Editor Insert Render Options Tools Window Help

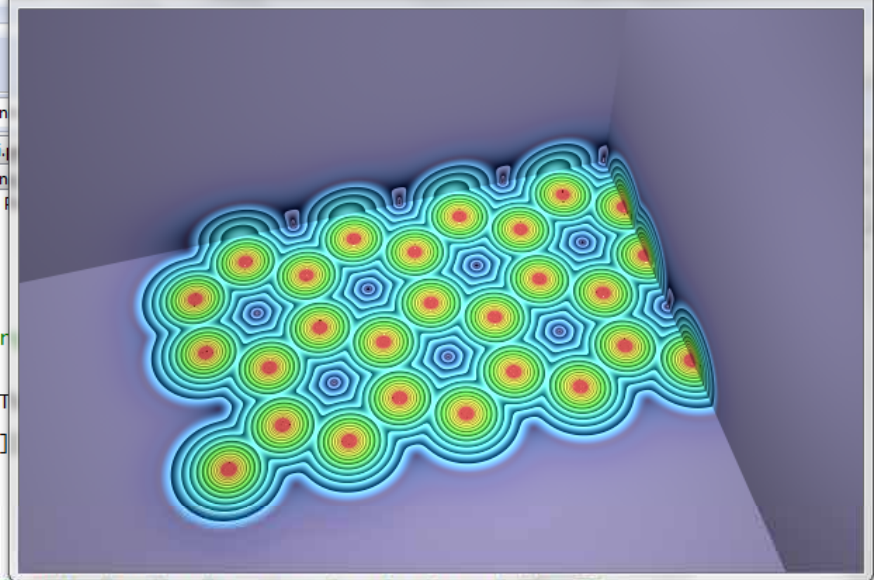
New Open Save Close Queue Hide Ini Sel-Run Run Pause Tray

[320x240, No AA] +Q11 +UA +W600 +H400 +RP5 +A +AM2 +R5 +FN8 +Omipro-is-great-Opatija2015 Declare=Varian

Messages | Histogram.inc | textures.inc | bk-TestyTekstur.pov | BoxWiskersy.pov | Test.pov | TlenBalloons-mini.
PolerowanyHistgram2.pov | MiPro-Polator-H2O.pov | MonoPatch.pov | BarShapes.in
texttester.pov | bk-UncertaintySTL.pov | STL.inc | STL.pov | Cameras.inc | bk-FuzzySurface.pov | scFinish.inc

```
  #end
  )
}
#end // CreateVF()

#macro CreateVFC2(_PTs,ff) // Coulombean potential from network of electric char

function(x,y,z){ // use: #declare VVV = CreateVFC( table)
  ff*qeff*(_PTs[1][0]/sqrt(pow(x-_PTs[1][1],2) + pow(y-_PTs[1][2],2) + pow(z-_PT
  #for(i, 2, _PTs[0][0])
     + _PTs[i][0]/sqrt(pow(x-_PTs[i][1],2) + pow(y-_PTs[i][2],2) + pow(z-_PTs[i]
  #end
  )
}
#end // CreateVF()


#macro CreateVLJ(_PTs) // create Lenard-Jones potential function from network of neutral molecules given by structure table _PTs:
                       // Q (ignored), x,y,z, radius [n][4], L-J radius r_m [n][5], potential well [n][6]
    #if (_PTs[0][0]>0)        // need one charge at least!
        function(x,y,z){
          _PTs[1][6]*( pow( _PTs[1][5] *_PTs[1][5]/(pow(x-_PTs[1][1],2) + pow(y-_PTs[1][2],2) + pow(z-_PTs[1][3],2)), 6)-2*pow( _PTs[1][5]*_PTs[1]

          #for(i, 2, _PTs[0][0])
              +  _PTs[i][6]*(pow( _PTs[i][5]*_PTs[i][5]/(pow(x-_PTs[i][1],2) + pow(y-_PTs[i][2],2) + pow(z-_PTs[i][3],2)), 6)-2*pow( _PTs[i][5]*_P
          #end
          }
    #else
        #declare VVV = function(x,y,z){0} // just for safety
    #end
#end // CreateVF()


#macro CreateVLJ2(_PTs,ff) // create Lenard-Jones potential function from network of neutral molecules given by structure table _PTs:
                       // Q (ignored), x,y,z, radius [n][4], L-J radius r_m [n][5], potential well [n][6]
    #if (_PTs[0][0]>0)        // need one charge at least!
```

590,325 [0.985,0.815]



Output -> 'mipro-is-great-Opatija2015.png' 8MB L:510 C:26 Ins 248 PPS 0d 00h 16m 07s

2a.Choice of scalar field

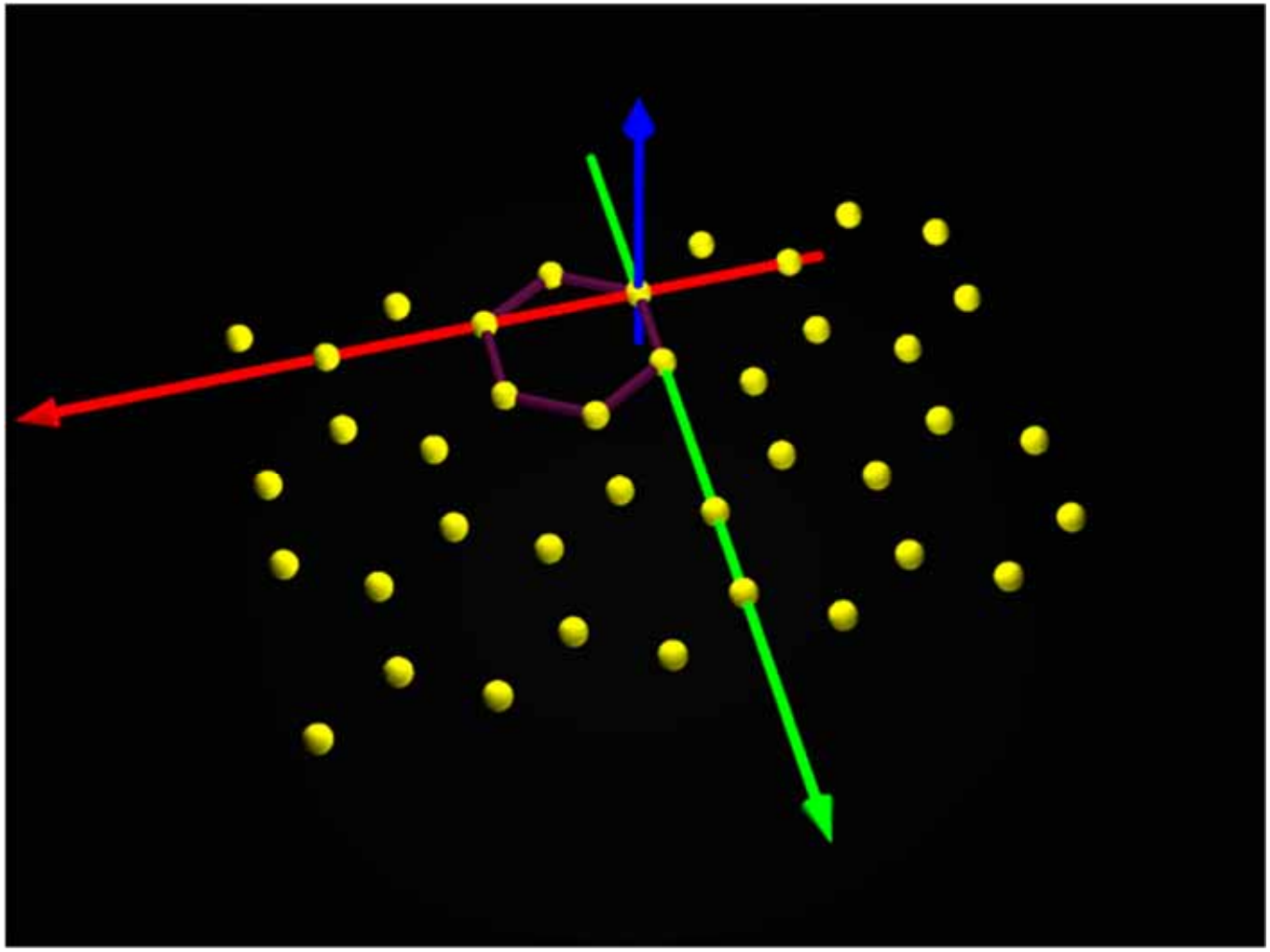
To compare different visualization styles the same scalar field produced by selected molecular structure should be used for all tested styles.

The structure chosen should obey at least two, contradictory, constraints:

1) should be complex enough to reveal hidden properties.

2) has to be as simple as possible to be computable in reasonable time on regular workstation.

2b.Graphene like flake



2b.Choice of scalar field

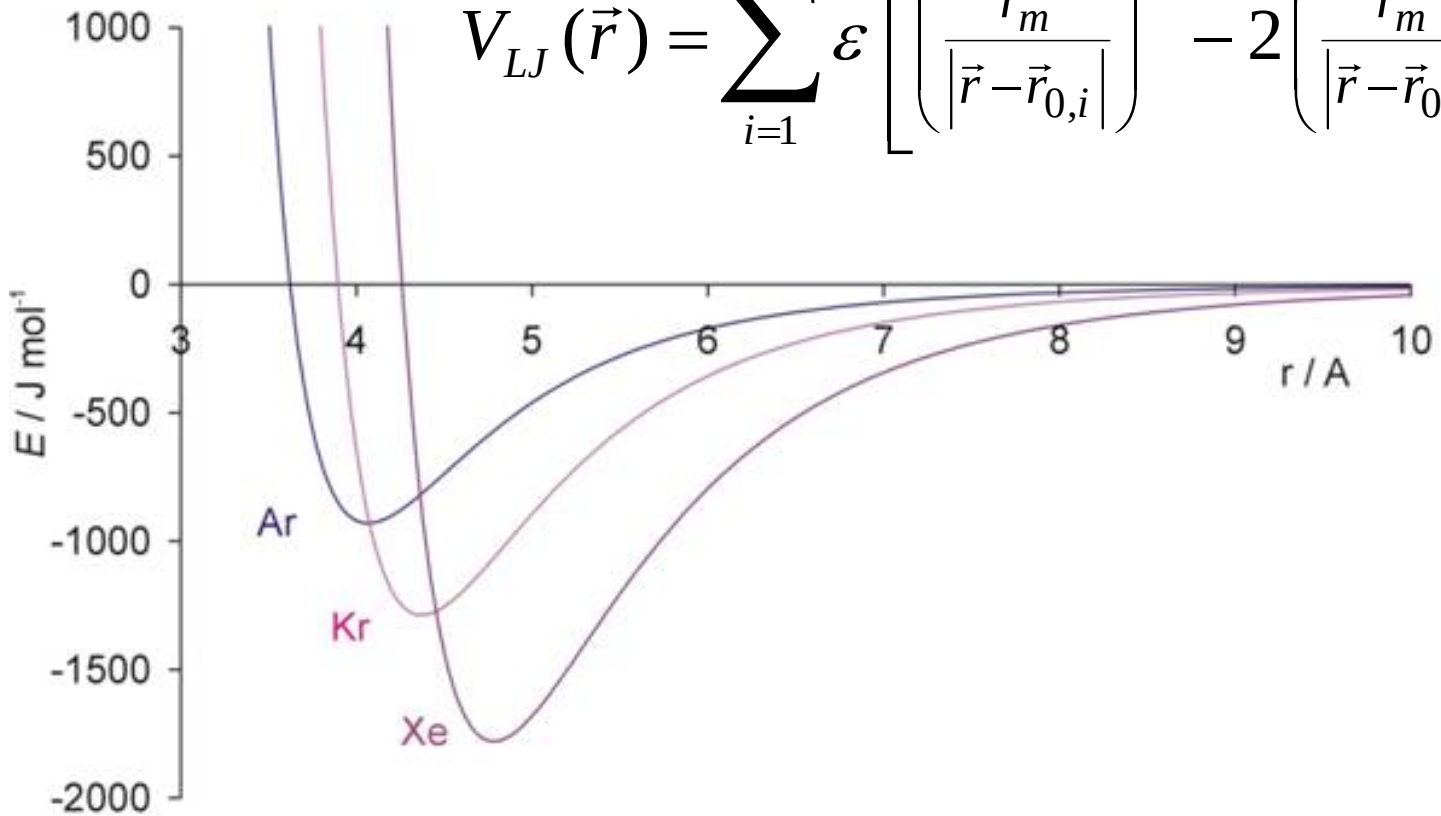
Lennard-Jones potential (L-J potential) around a single layer hexagonal structure (thus referred in the paper as "*graphene like*") composed from 40 neutral molecules, has been chosen.

It should be stressed, that electrostatic potential around real graphene flake is generated by electric charge distributed continuously due to the principles of the quantum mechanics and should be computed by integration over space using Coulomb's Law.

Conclusions remain the same for any static scalar field, e.g. tissue density restored from NMR or X Ray DICOM images.

3. L-J „potential”.

$$V_{LJ}(\vec{r}) = \sum_{i=1}^{N} \varepsilon \left[\left(\frac{r_m}{\left|\vec{r} - \vec{r}_{0,i}\right|} \right)^{12} - 2 \left(\frac{r_m}{\left|\vec{r} - \vec{r}_{0,i}\right|} \right)^{6} \right]$$



http://www.chm.bris.ac.uk/~chdms/Teaching/Chemical_Interactions/page_05.htm

4. Computer

Most calculations were done on ASUS 1215N, which is rather slow, if compared to high-end workstations.

Computation time on multicore computer, in comparison to obtained times, was several times shorter (AMD, quad core, 2.6 GHz, 8GB RAM).

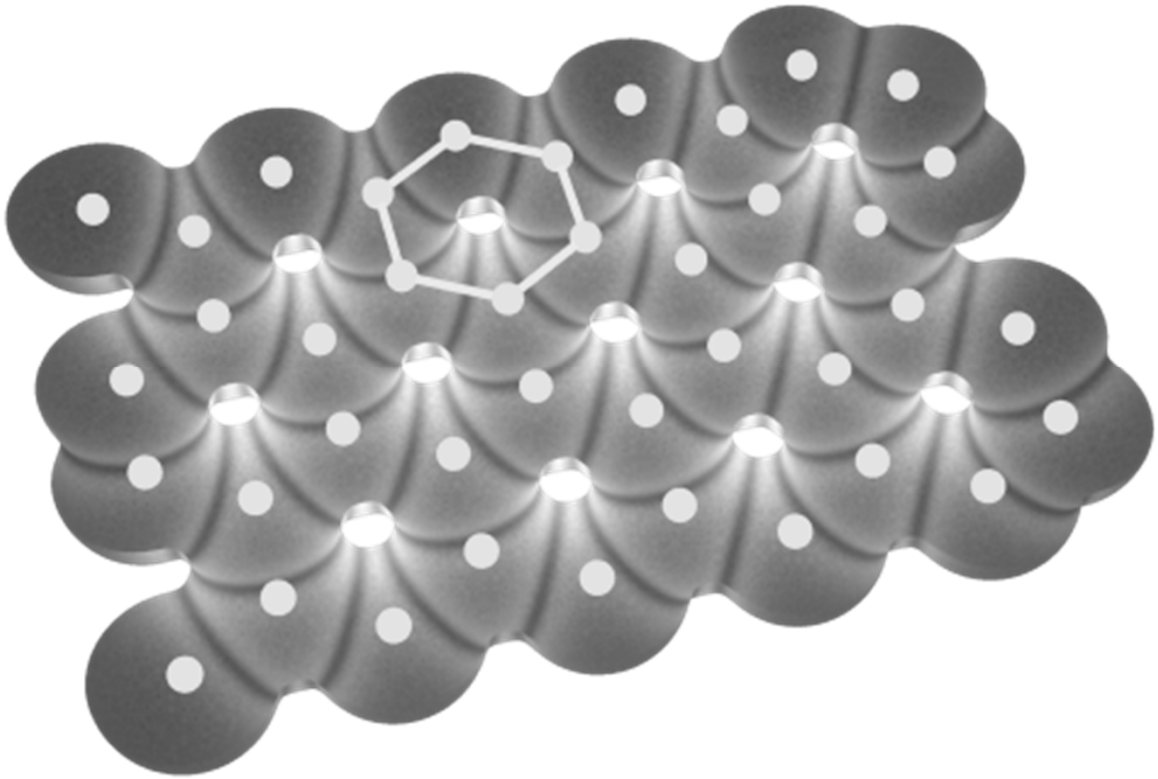
Unfortunately, there is no POVRay fork able to utilize CUDA technology however commercial GPU renderers are available (ex.FurryBall).

Style: Isosurface

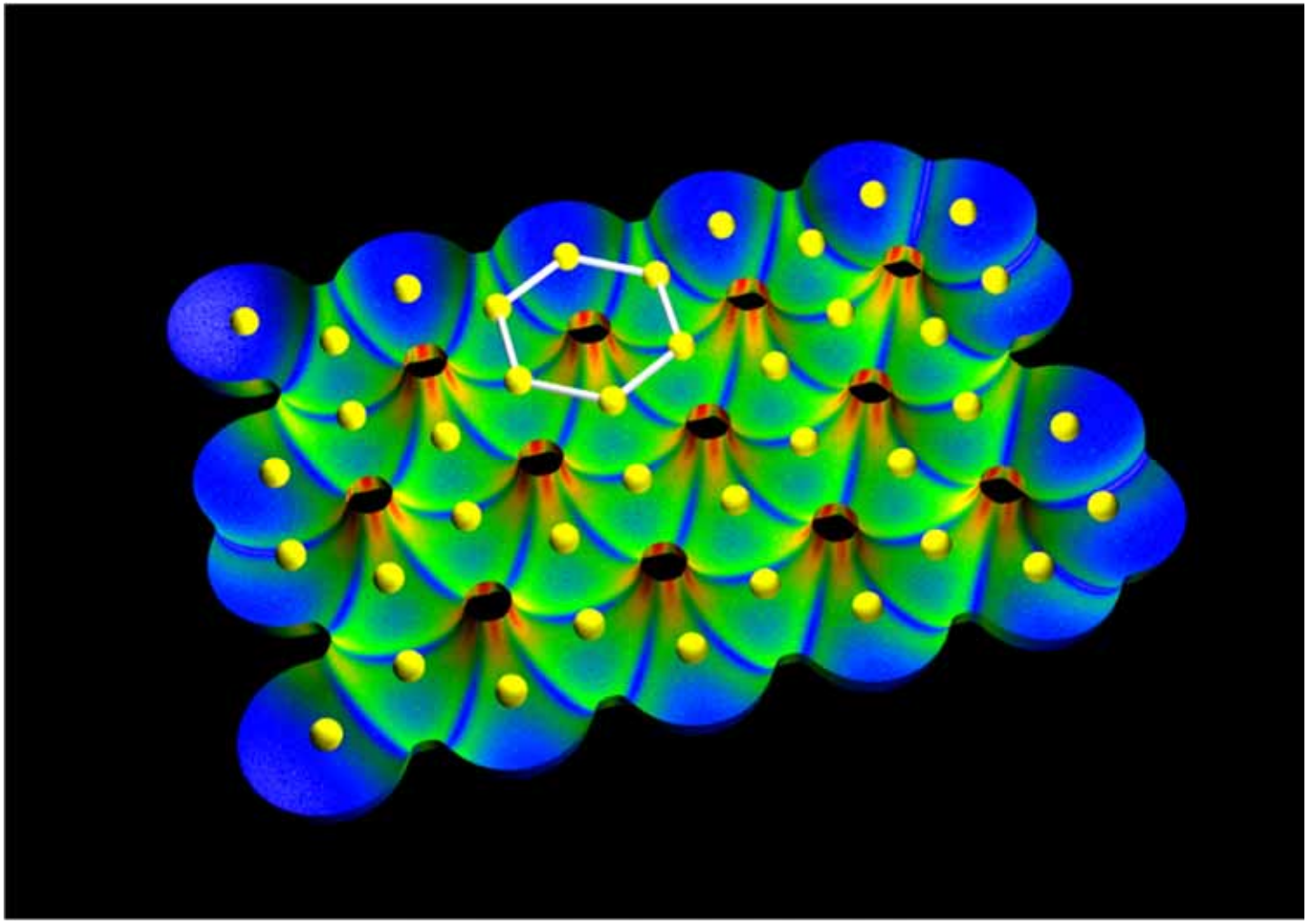
```
isosurface {
  function { FUNCTION_ITEMS }
  [contained_by { SPHERE | BOX
}]
  [threshold FLOAT_VALUE]
  [accuracy FLOAT_VALUE]
  [max_gradient FLOAT_VALUE]
  [evaluate P0, P1, P2]
  [open]
  [max_trace INTEGER] |
       [all_intersections]
  [OBJECT_MODIFIERS...]
}
```

```
#local grV0 = fn_Gradient(VLJ);
#local grV = function{
    clip(grV0(x,y,z)-gMin, 0, gMax-
            gMin)/(gMax-gMin)
};
SetIsoTexture(
  texture{
      pigment{
        function{ grV(x,y,z) }
        color_map{ _ColorMap }
      }
      finish{ Dull }
  }
)
#declare ISO3 = MakeEquiPlane(
VVV, 5, Min, Max )
object{ISO3}
```

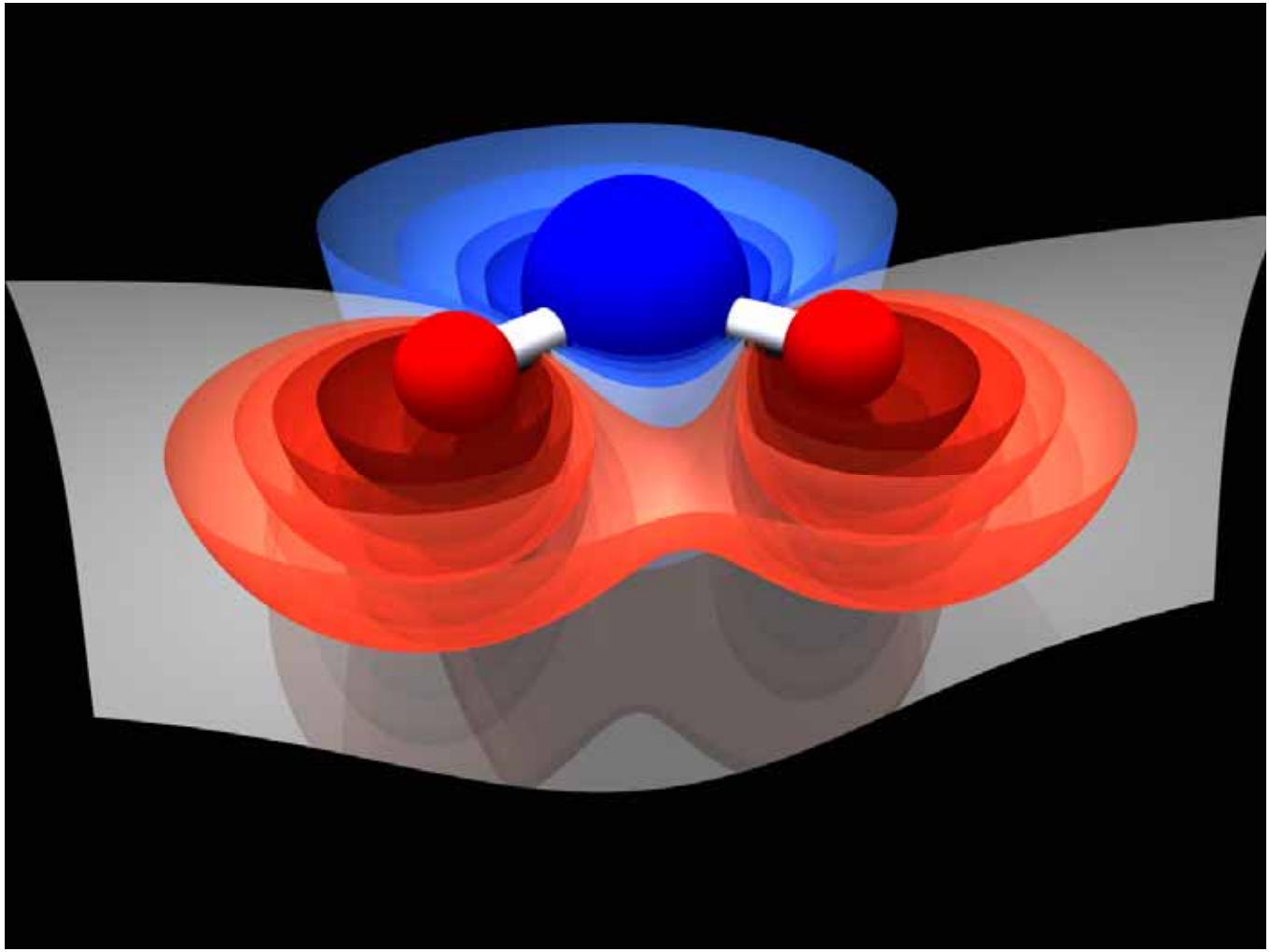
Style: Isosurface



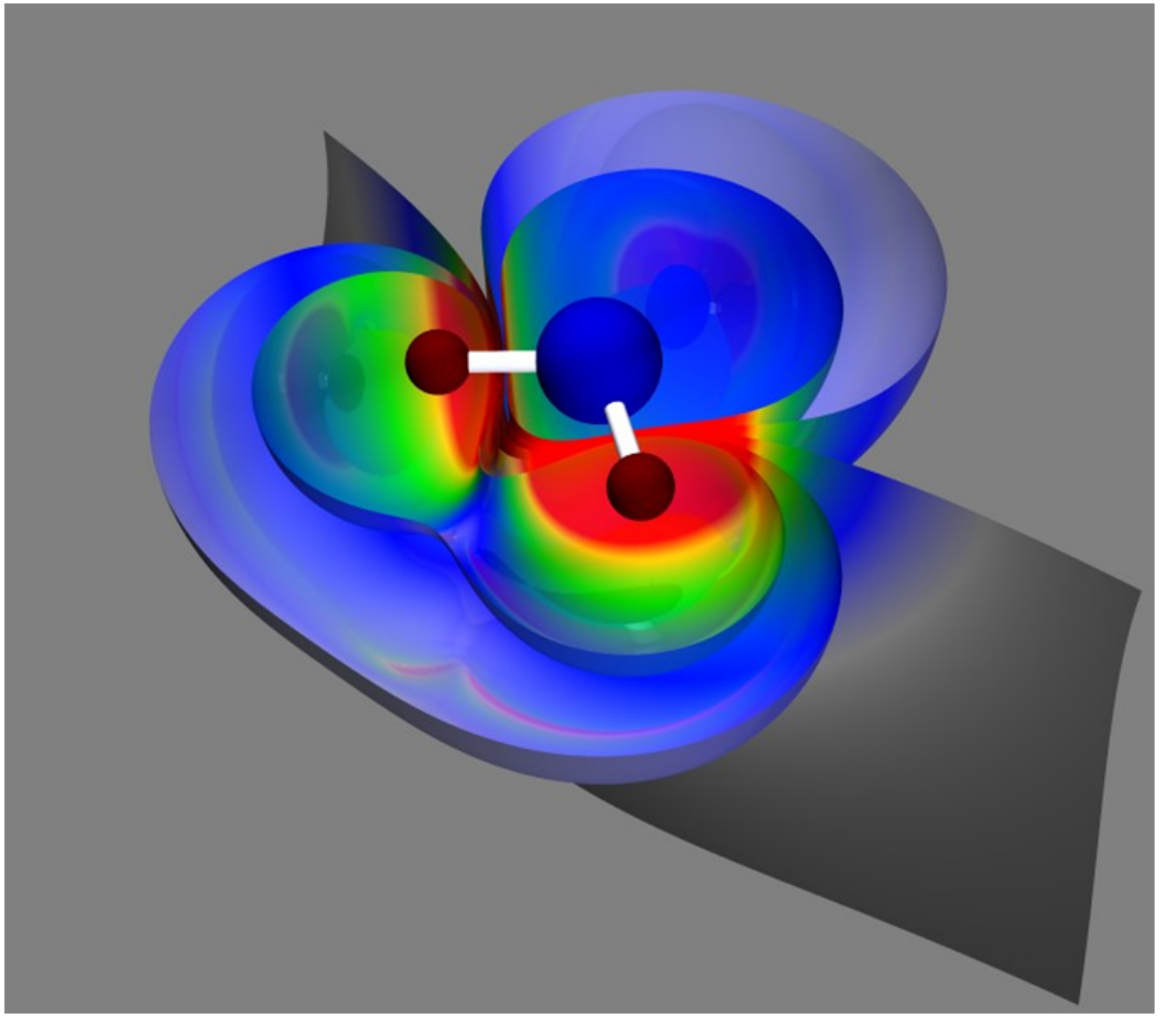
Style: Isosurface



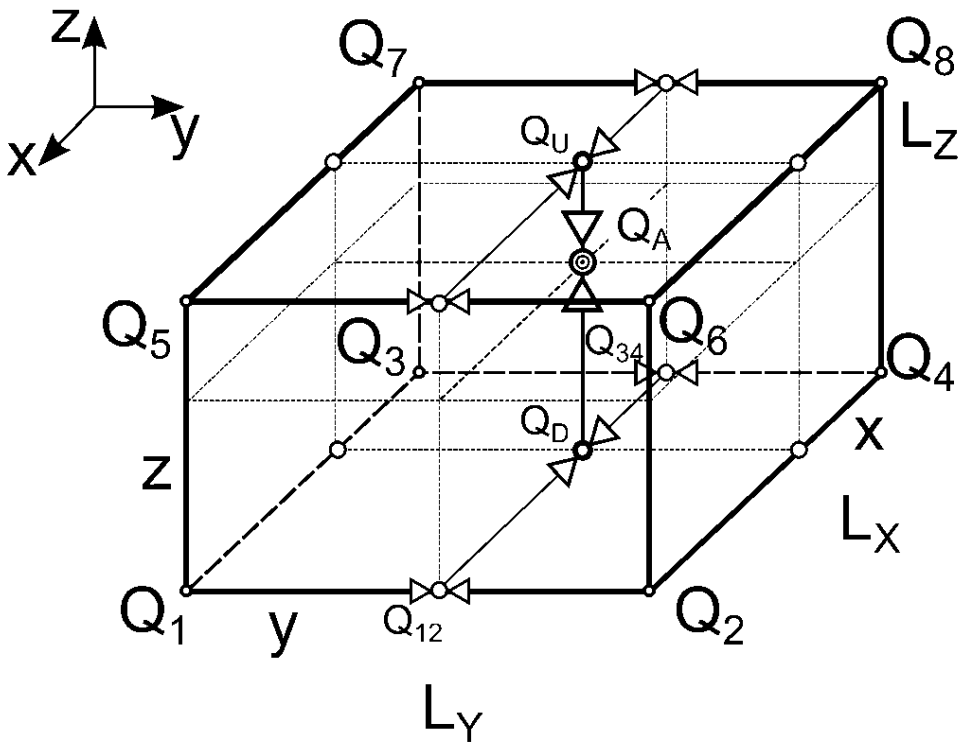
Style: Isosurface



Style: Isosurface, H_2O electrostatic field



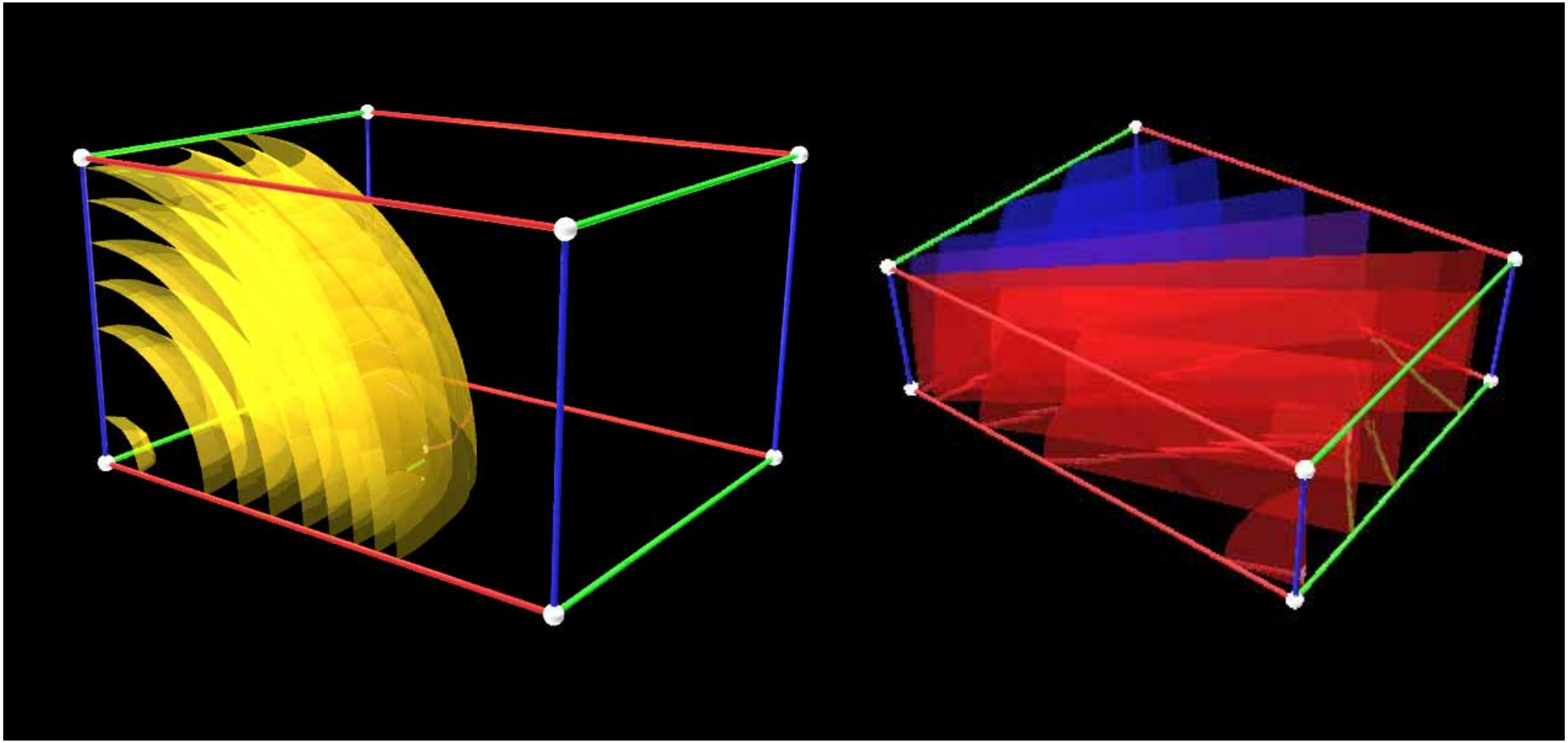
Style: Cellular trilinear interpolation



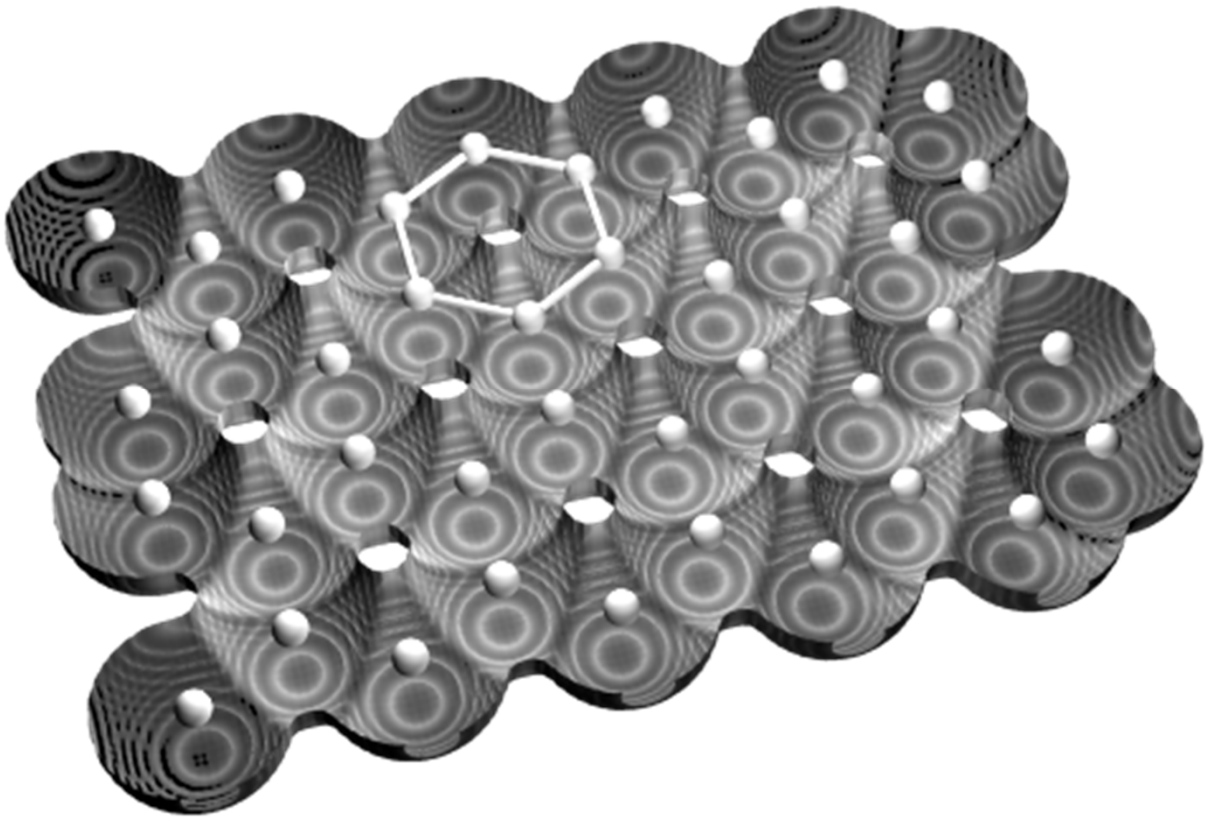
$$w(i,\lambda_x)=\begin{cases}1-\lambda_x & ,i=0\\ \lambda_x & ,i=1\end{cases}$$

$$Q(\lambda_x,\lambda_y,\lambda_z)=\sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1}w(i,\lambda_x)w(j,\lambda_y)w(k,\lambda_z)\cdot Q_{ijk}$$

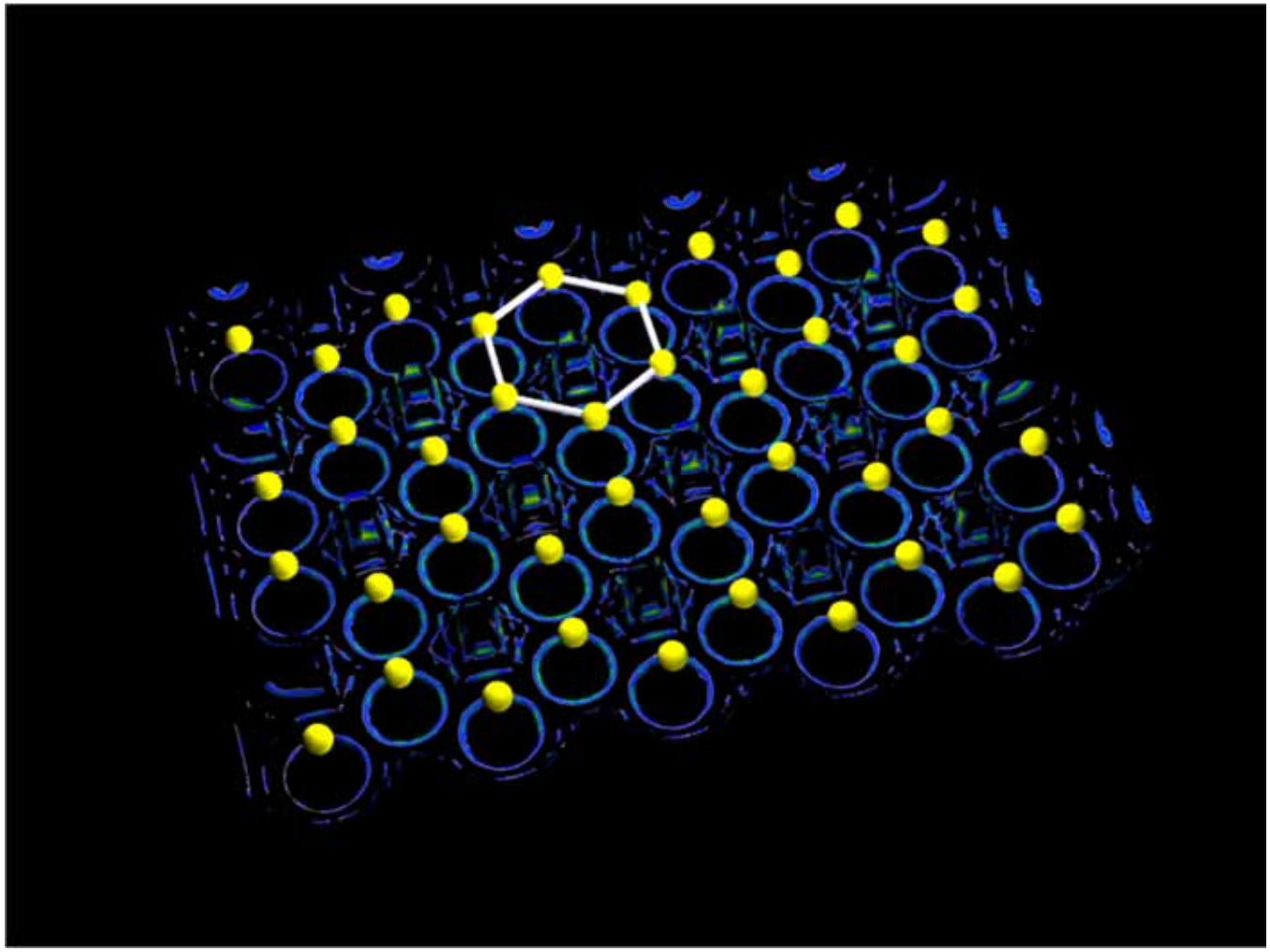
Style: Cellular trilinear interpolation



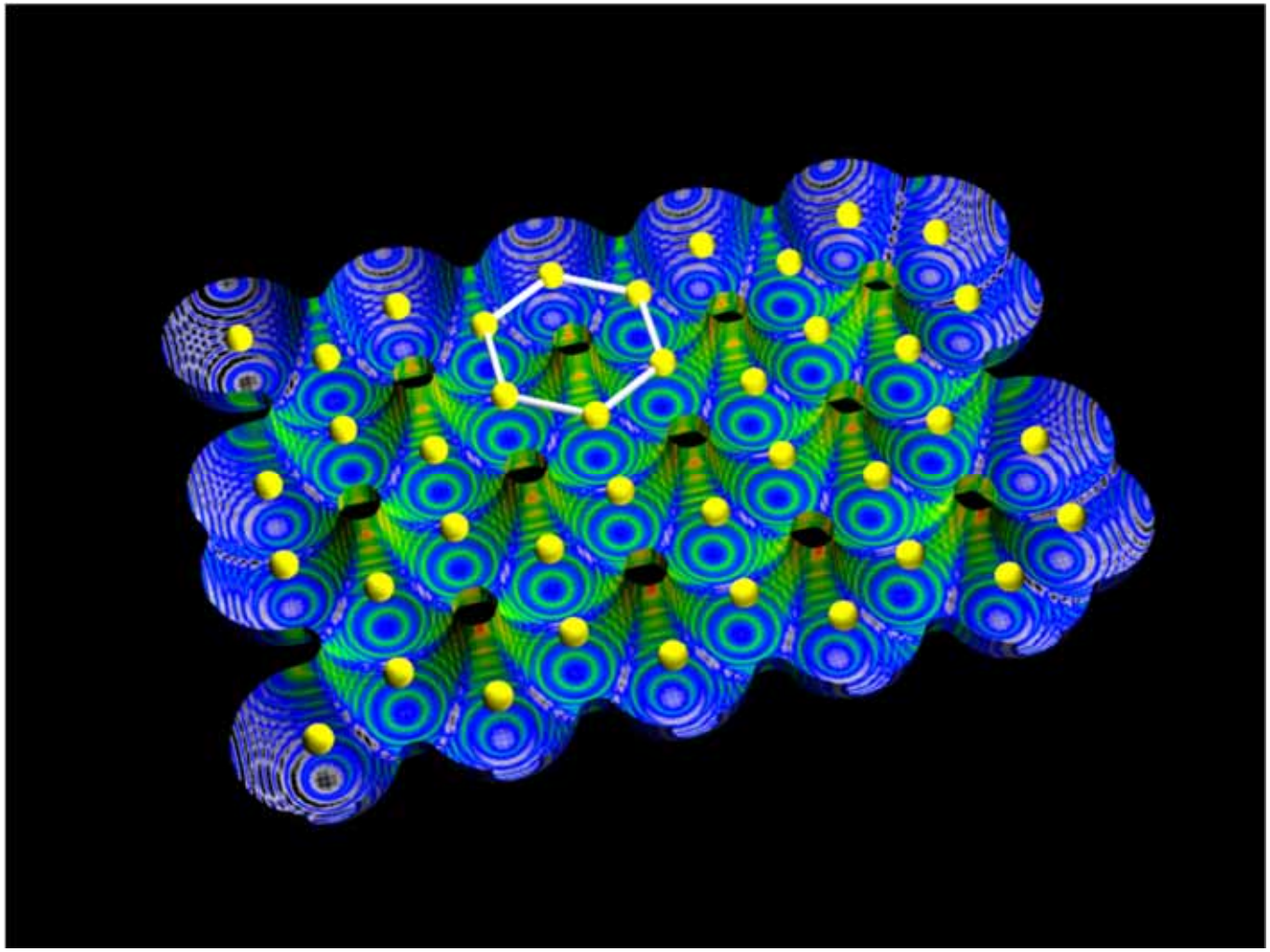
Style: Cellular trilinear interpolation



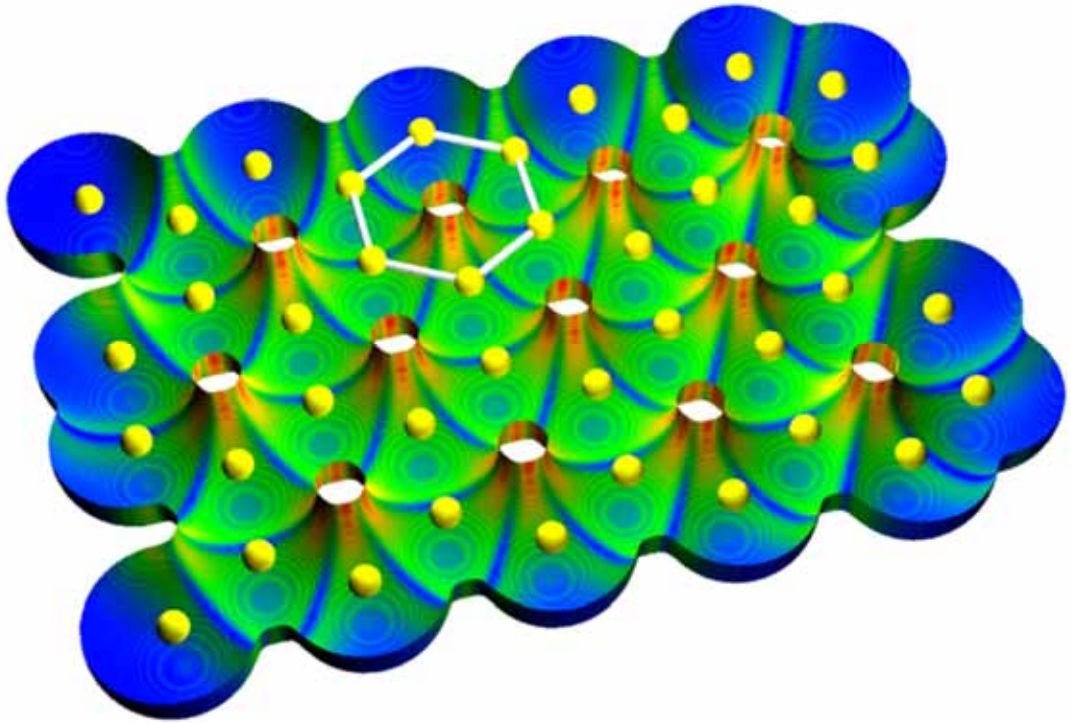
Style: Cellular trilinear interpolation



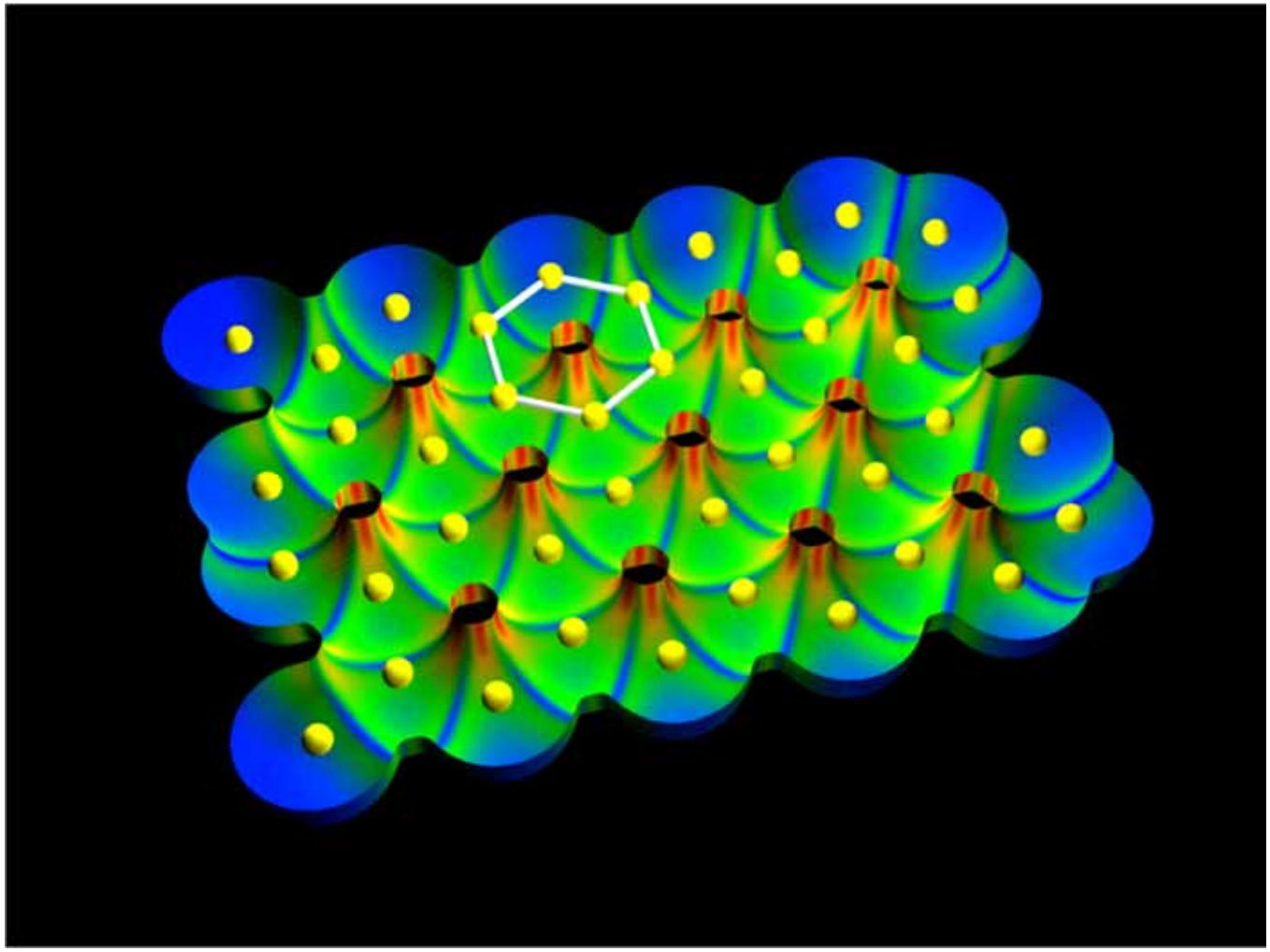
Style: Cellular trilinear interpolation



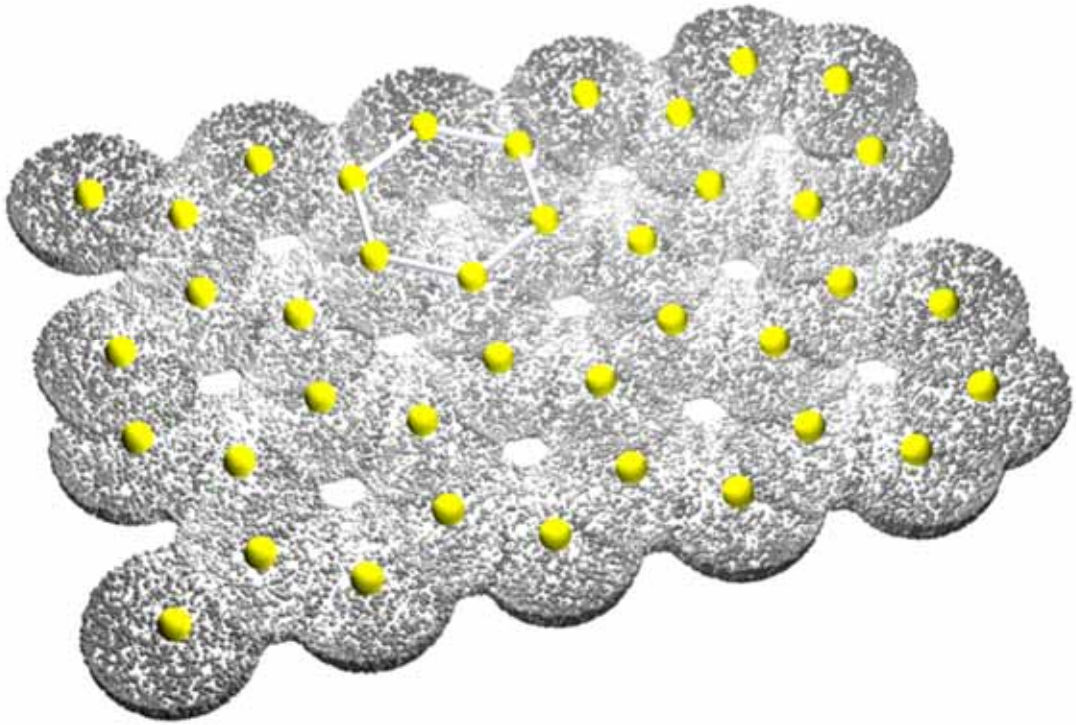
Style: Cellular trilinear interpolation



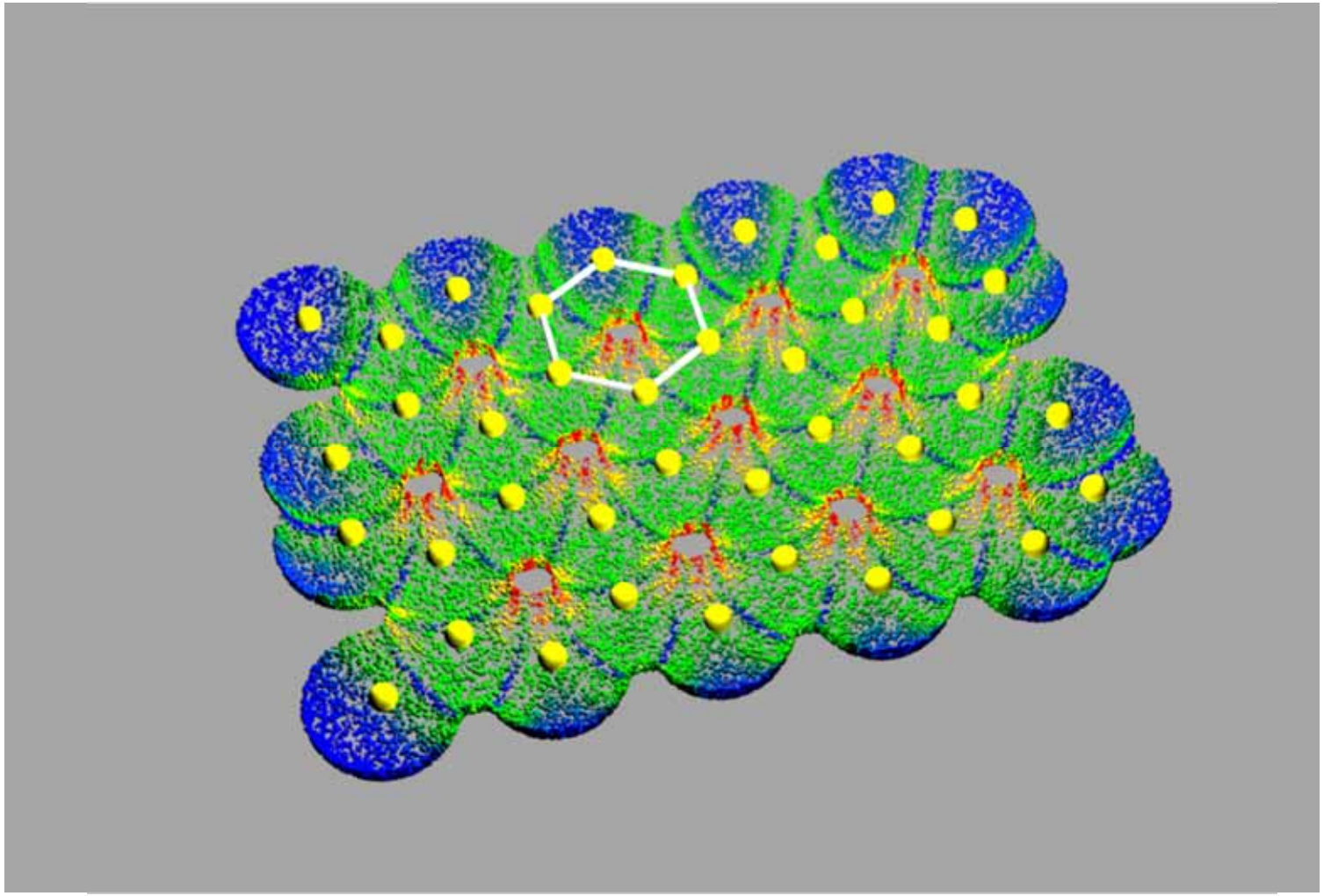
Style: Cellular trilinear interpolation



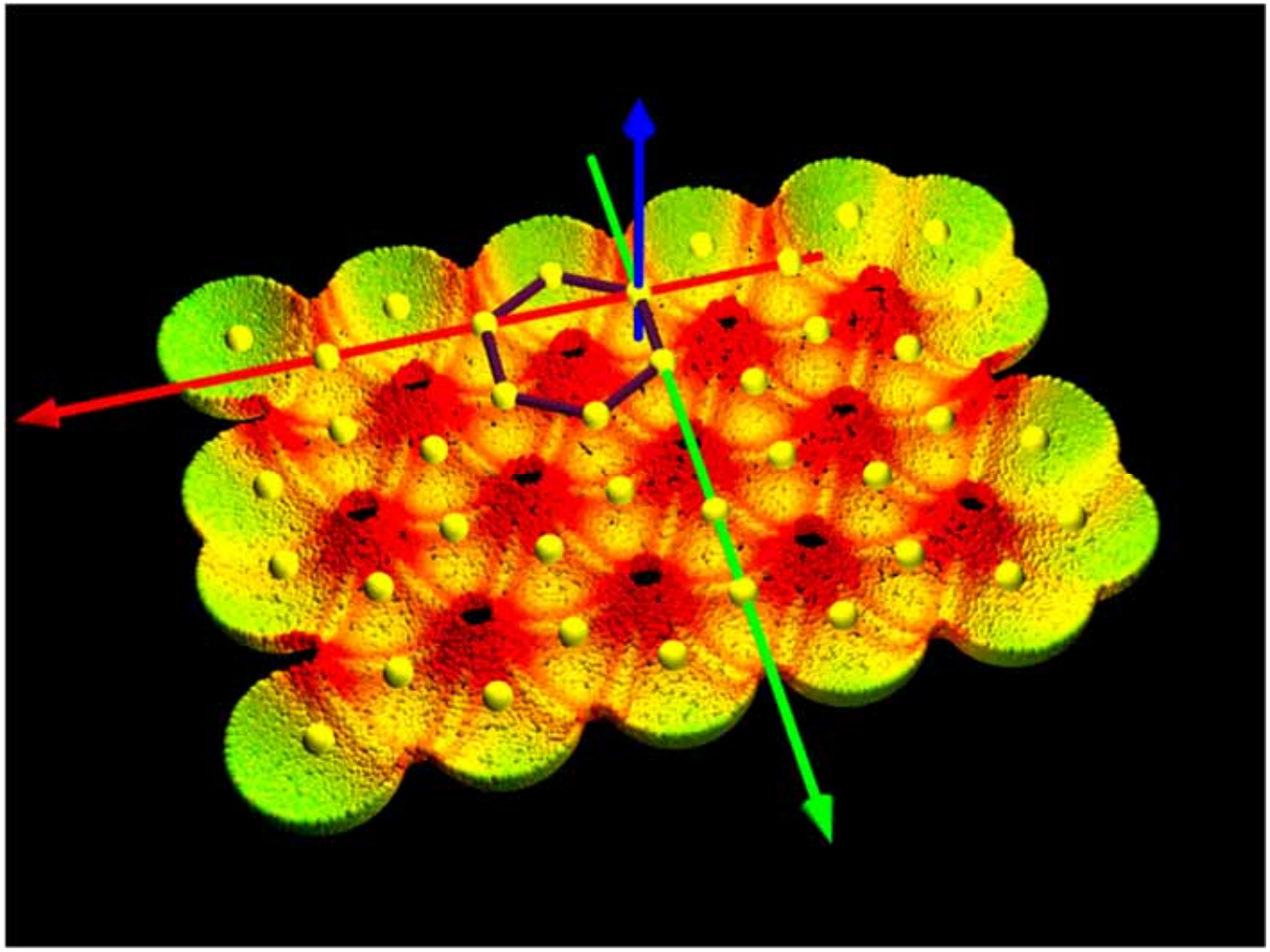
Style: Pseudoparticles approach



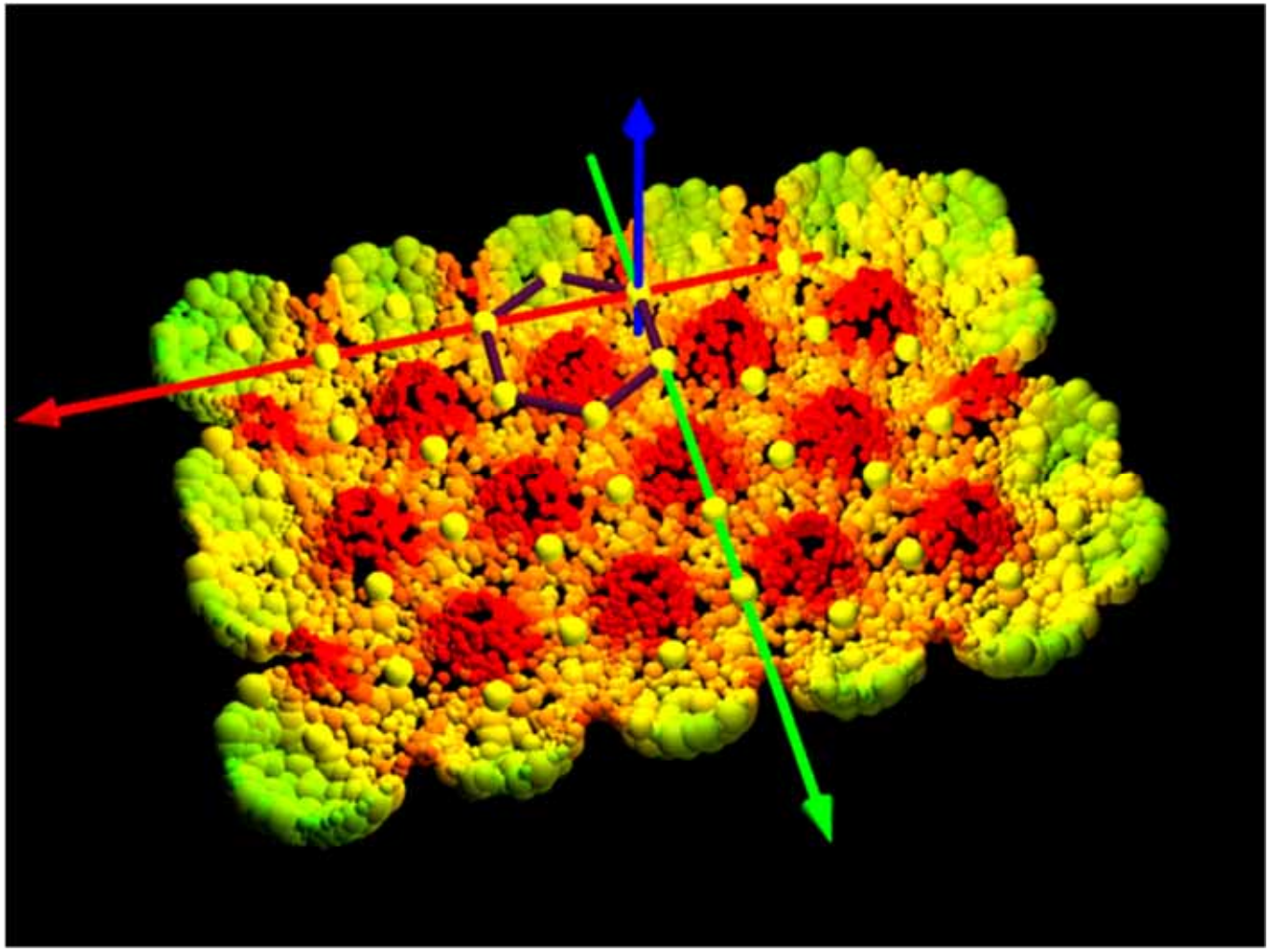
Style: Pseudoparticles approach



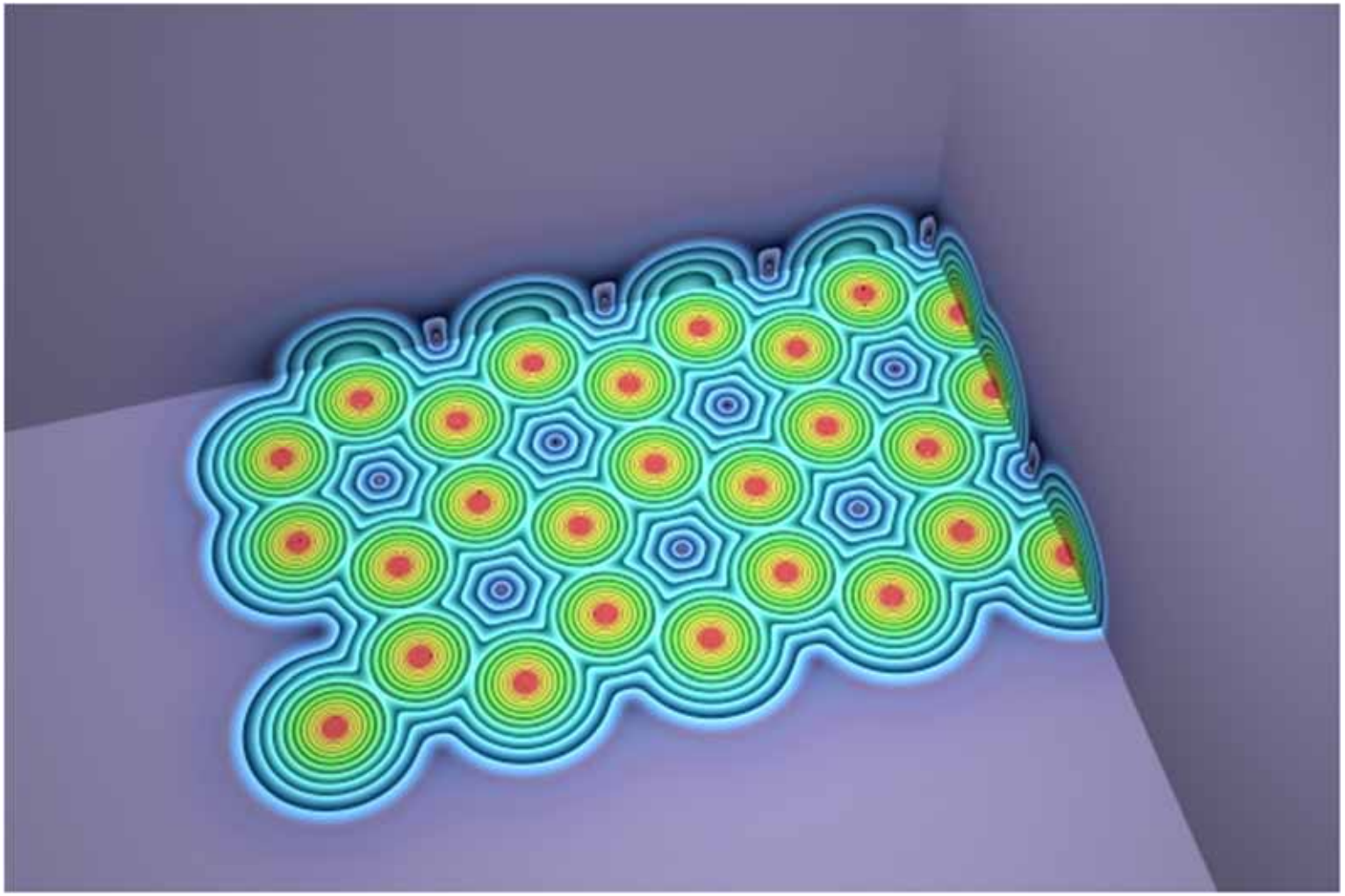
Style: Pseudoparticles approach



Style: Pseudoparticles approach



Style: Color maps, layered textures



Final remarks

High quality 3D render program equipped with efficient scripting language and supplementary API, like POV Ray, may be used for fast prototyping of complex visualization styles.

Because of numerical efficiency problems in some cases external program should be used.

Due to relatively short computation time color map style is very well suited for fast introductory analysis of new structures.

Thank you for your attention!