

# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ  
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY  
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

## VISUALIZATION AND MODELLING OF MOLECULES AND CRYSTALLES

DIPLOMOVÁ PRÁCE

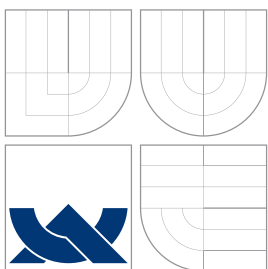
MASTER'S THESIS

AUTOR PRÁCE

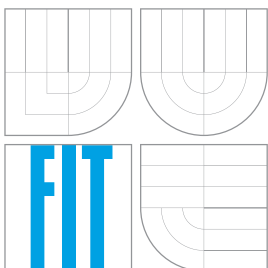
AUTHOR

VÁCLAV BUBNÍK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ  
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ  
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY  
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

## VIZUALIZACE A MODELOVÁNÍ MOLEKUL A KRYSTALŮ

VISUALIZATION AND MODELLING OF MOLECULES AND CRYSTALS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

VÁCLAV BUBNÍK

VEDOUCÍ PRÁCE

SUPERVISOR

Doc. Dr. Ing. PAVEL ZEMČÍK

KONZULTANT

CONSULTANT

SIXTEN BOECK

BRNO 2008

## **Abstrakt**

Aplikace pro vizualizaci a modelování molekul nejsou dosud příliš poznamenány současným hardware vyvinutým pro potřeby počítačových her. Cílem projektu je navrhnout intuitivní rozhraní s novými widgety specializovanými na atomové struktury a vizualizací využívající moderní hardware grafických karet. Důležitou částí je také dosažení vysoké přesnosti modelování, obvykle dostupné pouze u profesionálních CAD programů.

## **Klíčová slova**

Počítačová grafika, vizualizace molekul, interaktivní modelování, widgety, postscript, splatting, volume rendering

## **Abstract**

Applications for visualization and modelling of molecules are not using the full potential of modern graphics hardware developed for demanding needs of computer games. The goal of this project is to design an intuitive interface including new widgets specialized to atomic structure editing with visualization which uses the hardware available in modern graphics cards. It is particularly focused on high precision of modeling, usually available only in professional CAD programs.

## **Keywords**

Computer graphics, visualization of molecules, interactive modeling, widgets, postscript, splatting, volume rendering

## **Citace**

Václav Bubník: Visualization and Modelling of Molecules and Crystals, diplomová práce, Brno, FIT VUT v Brně, 2008

# Visualization and Modelling of Molecules and Crystals

## Prohlášení

Prohlašuji, že jsem tuto semestrální práci vypracoval samostatně pod vedením pana Zemčička. Další informace mi poskytl Sixten Boeck. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....  
Václav Bubník  
May 17, 2008

© Václav Bubník, 2008.

*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
<b>2</b>	<b>Existing molecular editors</b>	<b>8</b>
2.1	JMolEditor . . . . .	8
2.2	Gabedit . . . . .	8
2.3	Chimera . . . . .	9
2.4	VMD . . . . .	9
2.5	MolWorks . . . . .	10
2.6	Pymol . . . . .	11
2.7	Materials Studio . . . . .	11
2.8	Crystalmaker . . . . .	12
<b>3</b>	<b>Widgets</b>	<b>13</b>
3.1	VTK . . . . .	13
3.2	3ds Max . . . . .	14
<b>4</b>	<b>Analysis</b>	<b>16</b>
<b>5</b>	<b>Solution</b>	<b>18</b>
5.1	Basic structures . . . . .	18
5.2	Atomic structure . . . . .	19
5.3	Visualization . . . . .	21
5.3.1	Perception of 3D . . . . .	22
5.3.2	Scene graph . . . . .	23
5.3.3	Renderer . . . . .	23
5.3.4	OpenGL manager . . . . .	23
5.3.5	Postscript output . . . . .	24
5.3.6	Point node . . . . .	25
5.3.7	Line node . . . . .	25
5.3.8	Text node . . . . .	25
5.3.9	Splat node . . . . .	26
5.3.10	Sphere node . . . . .	28
5.3.11	Bond node . . . . .	29
5.4	Optimization . . . . .	30
5.4.1	Octree . . . . .	30
5.4.2	Level of detail . . . . .	31
5.5	Interaction . . . . .	32
5.5.1	Precision . . . . .	32

5.5.2	Transformations	32
5.5.3	User experience	33
5.5.4	Picking	34
5.5.5	Snapping	34
5.5.6	Graphic user interface	35
5.5.7	Toolbars	35
5.5.8	Icons	36
5.5.9	Dock	37
5.6	Gizmos	37
5.6.1	Sphere gizmo	38
5.6.2	Picker gizmo	39
5.6.3	Plane gizmo	39
5.6.4	Translation gizmo	40
5.6.5	Rotation gizmo	41
5.6.6	Image map gizmo	42
5.6.7	Array gizmo	44
5.6.8	Snap point gizmo	46
5.6.9	Measure gizmo	46
5.6.10	Bonds gizmo	47
5.6.11	Construction plane gizmo	48
<b>6</b>	<b>Experiments</b>	<b>49</b>
6.1	Setup	49
6.1.1	Qt, VTK and GLUT	49
6.1.2	Structures	49
6.1.3	Input devices	50
6.2	Visualization	50
6.2.1	Splats	50
6.2.2	Spheres	52
6.2.3	Materials	55
6.2.4	Bonds	56
6.3	Postscript	57
6.4	Hardware requirements	58
<b>7</b>	<b>Conclusion</b>	<b>59</b>
	<b>Appendices</b>	<b>65</b>
.1	Reference machine	65
.2	CD	65
<b>A</b>	<b>Screenshots</b>	<b>66</b>
<b>B</b>	<b>Class Hierarchy</b>	<b>69</b>
<b>C</b>	<b>Atom Editor: User's guide</b>	<b>74</b>
C.1	Introduction	75
C.2	Application	75
C.3	Controls	76
C.3.1	Mouse	76

C.3.2	Keyboard	76
C.3.3	Motion tracking devices	77
C.4	GUI	78
C.4.1	Toolbar	78
C.4.2	Dock	82
C.4.3	Elements	90
C.5	Video tutorials	90

# List of Figures

2.1	VMD	9
2.2	MolWorks	10
2.3	MaterialsStudio	11
2.4	CrystalMaker	12
3.1	VTK widgets.	13
3.2	3ds Max viewports	14
3.3	Translation manipulator in 3ds Max	15
3.4	Rotation manipulator in 3ds Max	15
5.1	Atomic structure	19
5.2	Grid	20
5.3	Structures in visualization	21
5.4	Stereo views	22
5.5	OpenGL vs. GL2PS vs. proposed solution	24
5.6	Cylinder in Postscript commands	24
5.7	Text node	25
5.8	Representation of a character	25
5.9	Splat node	26
5.10	Splat	27
5.11	Splats	27
5.12	Rendering methods in the sphere node	28
5.13	Bond node	29
5.14	Multiple bonds	29
5.15	Octree	30
5.16	Level of detail	31
5.17	Snapping modes	34
5.18	Graphic user interface	35
5.19	3d icons	36
5.20	Dock	37
5.21	Elements	37
5.22	Sphere gizmo	38
5.23	Scaling gesture	38
5.24	Plane gizmo	39
5.25	Translation gizmo	40
5.26	Composition of translation gizmo	40
5.27	Direct manipulation	41
5.28	Rotation gizmo	41
5.29	Composition of rotation gizmo	42



5.30	Periodic table of elements	43
5.31	Alternative periodic table	43
5.32	The array gizmo	44
5.33	Composition of array gizmo	44
5.34	GaAs crystal	45
5.35	The snap point gizmo	46
5.36	The measure gizmo	46
5.37	Dihedral angles	47
5.38	Bond gizmo	47
5.39	Construction plane gizmo	48
6.1	Patriot demo	50
6.2	Splats	51
6.3	Spheres	52
6.4	Sphere test with real samples	53
6.5	Fill rate	54
6.6	Material test	55
6.7	Bond test	56
6.8	Postscript content	57
6.9	File size of the Postscript outputs	57
6.10	HW test	58
A.1	Selections	66
A.2	Perception of 3d	67
A.3	Postscript	68
C.1	Application window	75
C.2	Toolbar	78
C.3	Dock	83
C.4	Sphere gizmo	83
C.5	Plane gizmo	84
C.6	Translation gizmo	84
C.7	Rotation gizmo	85
C.8	Array gizmo	86
C.9	Snap point gizmo	87
C.10	Measure gizmo	87
C.11	Dihedral angles	88
C.12	Bonds gizmo	88
C.13	Construction plane	89
C.14	Elements	90

# Chapter 1

## Introduction

Computers become common. People use them to solve problems from simple tasks to complex procedures and some tasks can be accomplished only with computers. As the computers become more and more powerful, we can interact with them using graphical user interfaces. For most users, a computer is only an instrument, helping them to reach their own goals. This can be facilitated with a good interface design which makes it easy for users to tell the computer what they want to do.

The aim of this project is to design an intuitive interface for interactive modeling of molecules and crystals with visualization that uses modern graphics hardware which is very important in materials research, biochemistry or physical chemistry and pharmacy drug design.

The possibility of displaying the atomic structures is important to understand their properties and to gain insight into the data. The data representing the coordinates of the atoms have three-dimensional character. This will influence both visualization and manipulation.

The structures contain from few atoms up to several hundred thousand atoms in large data sets. Visualization combined with interactive modeling shows always the current state of edited structure. Thus it is necessary that the system is well scalable. The algorithms and the dataflow are further optimized for real time interaction to reach short responses to the user inputs.

The purpose is either to create completely new structures or to modify existing data. The source of input data are previous simulations and scans of real materials. The results can be used to create images for publications or as an input for upcoming simulations. A combination of a text editor or a script and some visualization tool is still commonly used for this purpose in the present time.

Chapter 2 is an overview of available, state-of-the-art applications for visualization and modeling of atomic structures, focusing on the user interfaces, approaches to editing, possibilities of visualization and supported input devices.

Chapter 3 shows the interfaces and different ways of manipulation from modeling programs and toolkits which influence this project, such as the visualization toolkit VTK and modeling tool 3ds Max.

Chapter 4 provides analysis and states the requirements for appropriate solution.

Chapter 5 describes presented solution together with the methods of implementation. The first part is oriented to visualization and acceleration techniques. Stereo rendering and other techniques used to increase perception of 3d are described briefly. The second part is about interactive manipulation with the objects using a set of widgets specialized on modeling of the atomic structures.

Chapter 6 describes few benchmarks performed during the project. The test with a motion capture device was the most important. The use of such instruments leads to a new form of user interaction.

The conclusion from presented applications describing their disadvantages is made in the last chapter. The solution how to realize a new approach to the area is presented.

This work is based on the previous Term project and Bachelor project.

## Chapter 2

# Existing molecular editors

The most of existing graphics applications used in chemistry, biology and physics are focused only to visualization of atomic structures. Structure editing is mostly missing or added later and thus nonintuitive. An overview of the current state of molecular editors with various ways of editing is presented.

The applications for visualization and modeling of atomic structures can be divided to non-commercial available and commercial. The common attribute of free available applications is that they are multi-platform tools and the source codes can be obtained in general. The list of applications is not complete. Only the applications related to this work are presented.

### 2.1 JmolEditor

Java Molecular Editor (JmolEditor) is a program for displaying, analyzing, editing, converting and animating molecular systems [27].

The structures can be created by adding new atoms or predefined groups to the specified places. It is possible to move or remove an atom. All actions are driven from the dialogs combined with selections of the atoms in the visualization. The dialogs can be opened from the menu or the toolbar.

The application has a minimal set of possibilities except the high number of predefined groups.

### 2.2 Gabedit

Gabedit is a graphic user interface to computational packages like Gaussian or Molpro. The advanced "Molecule Builder" allows to rapidly sketch in molecules and examine them in 3D [25].

The editor contains a drawing area and a toolbar. The tools can be activated from the toolbar. For example there is a tool which creates the atoms. A new atom is placed under the mouse cursor with each mouse click when the tool is active. This can be compared to a paint brush tool in drawing applications. The other tools include selection, rotation and translation of the atoms and an eraser.

## 2.3 Chimera

Chimera is a program for interactive visualization and analysis of molecular structures and related data, including density maps, sequence alignments and trajectories. High quality images and animations can be generated [6].

The application consists of one window divided to a menu and OpenGL visualization. The visualization part is controlled with a mouse and a keyboard. The dialogs for various actions can be instantiated from the menu. For example to measure a distance between two atoms, it is necessary to open the dialog called Structure Measurements from menu-Tools-Structure Analysis-Distances. Holding the keys Control-Shift then to click on two atoms and in the dialog to choose the button Create. The other actions are as complicated. The advantage of this approach is easy extensibility and rich options in the dialogs.

The atomic structures can be created from scratch in Chimera. This is further simplified with a dialog that allows to replace one atom with one typical pattern containing more atoms.

## 2.4 VMD

VMD (Visual Molecular Dynamics) is molecular graphics program designed for the interactive visualization and analysis of biopolymers [28].

The application consists of three windows (*Fig. 2.1*) and a set of dialogs.

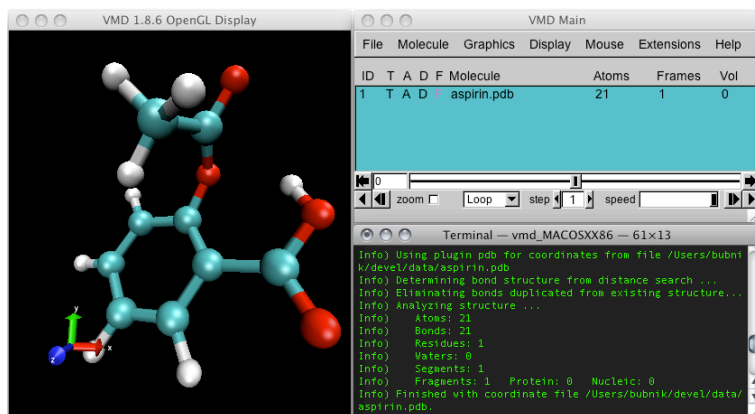


Figure 2.1: The user interface in VMD. The visualization window shows the molecule of aspirin.

VMD includes a script language which controls the whole application. The commands can be also entered separately to the terminal window. The other way of user interaction is to select the commands from the menu. The interaction in the visualization window uses the mouse combined with keyboard shortcuts. This action is complicated with several exclusive mouse modes like rotation, translation or selection (similarity with editing modes in vi text editor).

About 60 formats for import and export of the molecules are supported in VMD. There are 22 ways of how to visualize. The other types of visualizations can be made from their combinations.

VMD is mainly a visualization tool and the possibilities of editing are limited. The atoms can be moved in one of the mouse modes. The bonds can be removed or created in another mouse mode. However new structures can not be created from scratch.

Various input devices are supported in VMD. One of them is Phantom with six degrees of freedom and tactile feedback. VMD can be displayed in several stereo modes using polarized glasses, head mounted displays and the CAVE.

## 2.5 MolWorks

MolWorks is an integrated software tool for molecular design. MolWorks supports a molecular builder and pre/post interface to quantum mechanic calculation software, such as Gaussian, MOPAC and GAMESS [18].

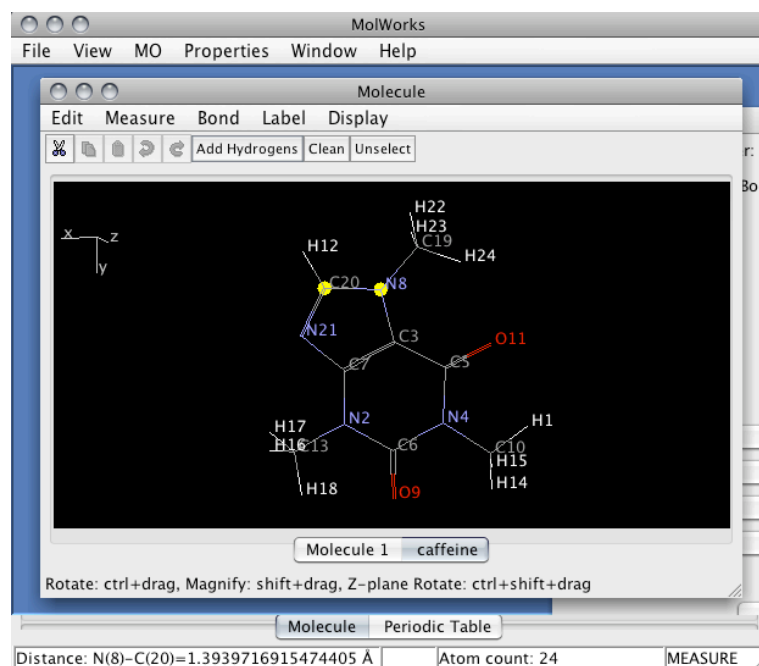


Figure 2.2: MolWorks application displaying a molecule of caffeine.

MolWorks uses multiple document interface (MDI). Multiple atomic structures, each in its own window, can be edited this way.

The application has limited visualization and editing options (*Fig. 2.2*). New atoms and bonds are created from drawn poly-lines where each mouse click adds an atom and connects it to the previous atom with a bond. The elements of the atoms can be changed. The atoms can be moved in the camera view plane or removed. It is also possible to measure the distance between two atoms.

The selections are divided to two modes. One for the atoms and the second for the bonds.

What makes this tool exceptional is that the order of the bonds is automatically computed and visualized with multiple lines. This feature can make the structures more readable.

## 2.6 Pymol

Pymol is a visualization and modeling software of atomic structures focused on high quality outputs for publications from biology [24] and creation of animations.

The application interface is created by a window with a menu. The window is divided to a command line, a list of existing objects in the scene and the main OpenGL visualization. Multiple level popup menus can be instantiated from the visualization and the list of objects. Various actions with the scene are controlled with these menus. Some actions require a mouse input together with specific combination of pressed keys on a keyboard.

Except the visualizations and measuring of distances it is possible to create completely new structures. This task is simplified with predefined patterns that can be bound to selected positions. A selection of atoms can be translated or rotated. The axis of rotation can be specified along selected bond.

Pymol supports three stereo modes including quad-buffer stereo for shutter glasses.

## 2.7 Materials Studio

Materials Studio is a software environment for visualization, simulation and modeling of the molecules, crystal materials and polymers [2]. The functionality depends on purchased modules. Only Materials Visualizer module will be described.

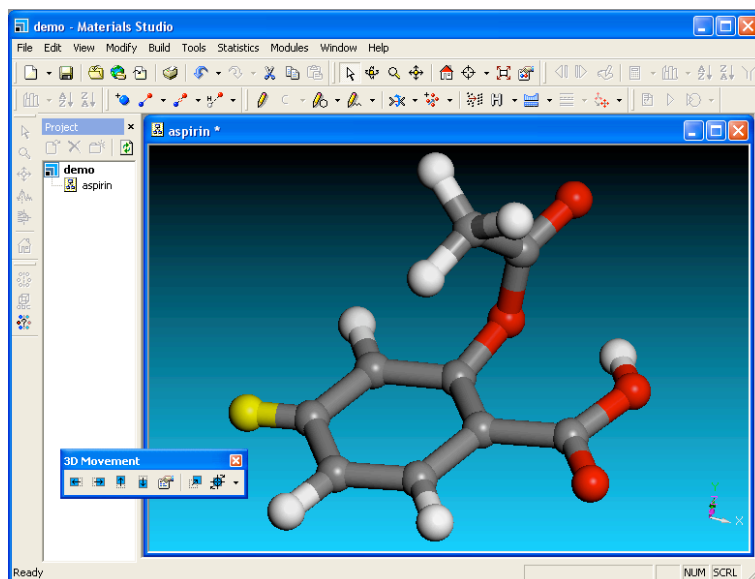


Figure 2.3: Materials Visualizer with the molecule of aspirin on gradient background.

The application uses multiple document interface with a menu and floating toolbars (Fig. 2.3).

New structures can be build with drawing like in Molworks. There is a high number of predefined structures that can be bind to existing atoms. An another way how to build a structure is to choose a symmetry from the crystal builder together with the parameters of the grids. It is possible to move or to remove the atoms or to measure distances and angles between them. The bonds can be created or removed.

All actions made by the user are recorded in the history with multiple undo/redo.

## 2.8 Crystallmaker

CrystalMaker is a program for building, displaying and manipulating all kinds of crystal and molecular structures [8]. CrystalMaker is commercial application.

The application has three main windows (*Fig. 2.4*). Multiple visualization windows can be opened, each for a single edited structure. The other windows contain the tools and a list of all atoms in the structure.

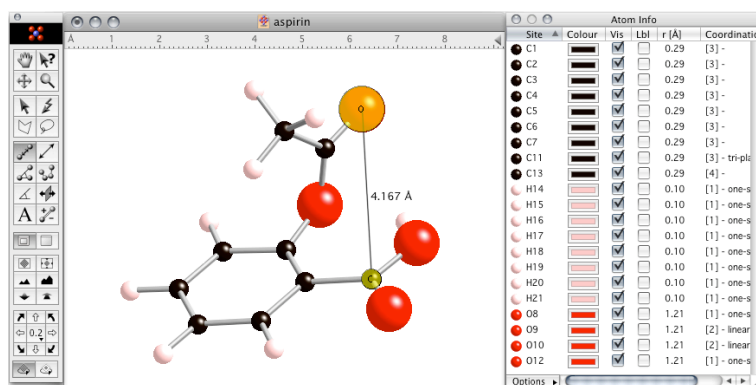


Figure 2.4: Three main windows in CrystalMaker. The visualization window shows the molecule of aspirin with one measured distance between two atoms.

The creation of a new structure differs for the molecules and the crystals. The molecules are set by definition of atomic coordinates. The crystals are started from predefined unit cells with various symmetries. New atoms can be added duplicating the existing ones or using the table.

The atoms can be moved, rotated or removed. It is possible to remove or to create the bonds and to measure the distances and the angles. These interactions depend on the active tool from the tool window. All operations are recorded with 99 levels of undo.

The visualization of the structures can be displayed in several styles with detailed options. CrystalMaker supports two stereo modes.



# Chapter 3

## Widgets

Common widgets like buttons, list boxes or text fields are well known interaction components in two dimensional interfaces. These elements can be used also in a 3d interface. Advanced modeling and interaction techniques exist out of the scope of existing atomic structure editors. An overlook about the current state is provided.

### 3.1 VTK

The Visualization Toolkit (VTK) is freely available software system for 3D computer graphics, image processing, and visualization [15]. VTK is more and more focusing on interaction with the data during the last years.

The widgets are the most important part of the toolkit in relation to this work. A widget is an object in a scene that responds to user events and data changes by corresponding changes in its appearance or behavior [1].

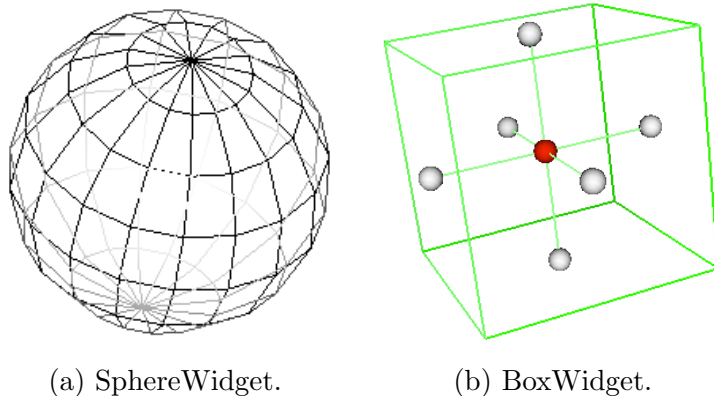


Figure 3.1: Two VTK widgets are shown on the image. The sphere widget (a) represents a location and a volume defined by the radius of the sphere. The box widget (b) controls the location, the scale and the orientation of an object in 3D space.

## SphereWidget

The sphere widget is visualized with wire model (*Fig. 3.1*) and represents two variables. The first variable is the location of the sphere, which can be modified with dragging (left mouse button). The second variable is the radius of the sphere, which can be also modified with dragging (right mouse button). The values can be then used from the different places in a program. The widget is suitable for selections of the atoms and other objects in the scene. Only the objects inside the sphere will be selected.

## BoxWidget

The box widget is visualized with wire model (*Fig. 3.1*) and seven spheres used as the handles. The handle in the center controls the location of the box. The other handles control the scale of the box. The orientation can be modified when a side of the box is picked and dragged.

The constraints are limited to three axes. For example the box can be scaled only along the x axis but the amount of scaling is not accurate. This widget was not found useful for selections after few experiments.

## 3.2 3ds Max

3ds Max is a 3D modeling, animation and rendering software [4]. It is designed for simple and intuitive modeling of objects created from polygon meshes.

The manipulators are basic editing elements. Each manipulator visually represents some actions through active parts which can be dragged with a mouse. The most commonly used manipulators are translation and rotation.

Multiple views to the scene can be used at the same time (*Fig. 3.2*). Horizontal projection (view from the top) and projection (view from the front) are some of possible views taken from descriptive geometry. Multiple views improve the orientation in 3D space.

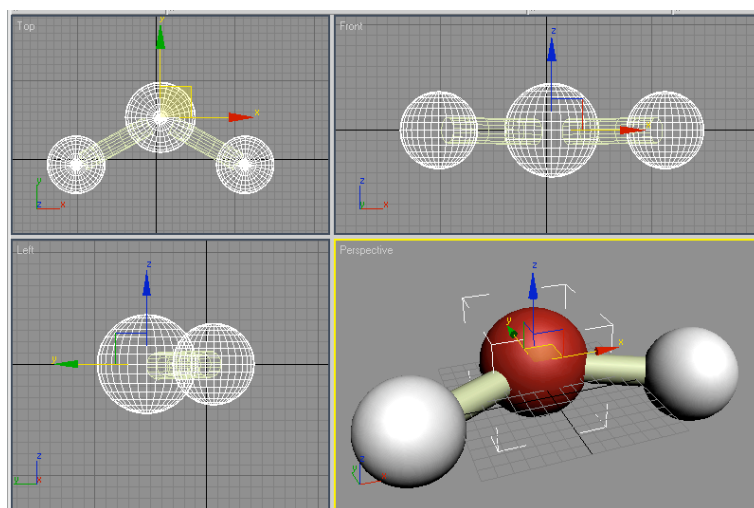


Figure 3.2: 3ds Max with multiple views to the scene: horizontal projection, projection, side and perspective projection.

## Translation

The manipulator for translations is represented with three arrows and three quads (*Fig. 3.3*). The arrows allow translation constrained to their axes. The quads allow translation constrained to their plains. All of these six elements provide visual feedback with changing the color when the mouse cursor is over their areas.

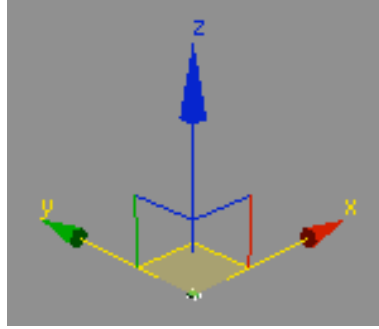


Figure 3.3: Translation manipulator from 3ds Max with three arrows corresponding to the axes of coordinate system.

## Rotation

The manipulator for rotations is represented with three arcs and two circles (*Fig. 3.4*). Three arcs lie in the planes perpendicular to the axes of coordinate system. The gray circle always lies in the view plane. The last circle is inactive. Selected objects can be rotated around the axes of the arcs. The center of rotation (pivot) is implicitly in the average of selected objects and can be optionally placed to specified coordinates.

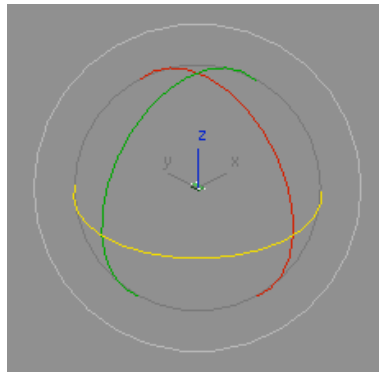


Figure 3.4: Rotation manipulator from 3ds Max with three arcs corresponding to the axes of coordinate system. The gray circle corresponds to the view plane.

## Chapter 4

# Analysis

There is no available molecular editor for intuitive and accurate editing of atomic structures. The concept of widgets as interactive elements in the scene has plenty of unexploited potentials to improve the user interface and to create a new ways of editing. The widgets allow both intuitive and accurate modeling features, but the current molecular editors are not using them (*Tab. 4.1*).

The essence of this work is to introduce the widgets to a modeling tool of atomic structures. The set of currently known widgets can be extended with new, more specialized widgets focused on interaction with the atoms.

The overall performance and development time of the system is determined by a good design at the lowest level. An efficient and safe memory representations of the data, especially of the atomic structure are required.

The atomic structures are composed from the atoms and the bonds. The bonds show interactions between the atoms. And as the atoms can be moved to arbitrary coordinates, it is necessary to update their bonds. The bond-searching algorithm responsible for the updates should be running at real-time speeds to provide a feedback during the interactions.

The system should be capable to provide visualization and interactions with high numbers of the atoms. Simple to setup and fast rendering pipeline is required in order to maintain thousands of atoms. The representations with high number of elements can benefit from organization to a spatial structure.

The quality of visual outputs is usually very high in available tools for visualization of molecules. This is accomplished with advanced lighting models like ambient occlusion in QUTEMOL [17], per-pixel Phong lighting from fragment programs in VMD or computed shadows [5]. Utilization of advanced rendering techniques and a low-level graphics library is required in order to create a competitive application. The rendering performance should be very fast to provide real-time responses to user inputs during the interactions with the scene.

Each editing tool has to have some sort of the output to save the work. An additional output can be realized with possibility of taking the screenshots.

The integration of the widgets leads to a 3d user interface. The other components like the buttons or the menus can be taken to 3d space as well. A virtual reality system for modeling of atomic structures can be created by adding a possibility of stereo visualization and a 3d interface which can be controlled by motion tracking input devices.

It should be possible to create constrained transformations of the atoms to reach accurate modification of the coordinates. This requirement is well suitable for the widgets which are able to visualize such constraints.

Table 4.1: The table provides an overview of available features in listed applications. The rows contains supported operating systems, the presence of advanced molecule builders, overall impressions of user interfaces, availability of a history of commands with undo and redo facilities, ratings of visual quality of the outputs including available settings, multiple bonds, sketching of new molecules with a mouse, availability of a terminal or a command line as additional interaction method and possibility of stereo views.

Support	Application							
	Chimera	Pymol	VMD	JMolEditor	Gabedit	Molworks	M. Studio	Crystallmaker
<b>OS</b>	Multi-platform	Multi-platform	Multi-platform	Multi-platform	Multi-platform	Multi-platform	Windows	Windows, Mac OS X
<b>Builder</b>	None	None	None	None	None	None	Good	Good
<b>Interface</b>	Complex	Normal	Complex	Low	Low	Low	Good	Good
<b>Undo</b>	Some	No	No	No	No	Some	Yes	99 levels
<b>Quality</b>	Normal	Normal	Rich Setup	Low	Low	Low	Normal	Good
<b>Multiple bonds</b>	No	No	No	No	No	Yes	Manual	No
<b>Measure Rotate Move</b>	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
<b>Sketching</b>	No	No	No	No	Yes	Yes	Yes	No
<b>Command line</b>	Yes	Yes	Yes	No	No	No	Scripts	No
<b>Stereo</b>	Yes	Yes	Yes	No	No	No	No	Yes

# Chapter 5

## Solution

A new approach in modeling of atomic structures and interaction with large data sets in general is presented in this chapter. The solution is based on effective representations of the data in memory, a scene graph based visualization using the modern graphics hardware, optimization spatial structures and specialized interactive elements focused on manipulation with atomic structures.

### 5.1 Basic structures

The implementation in C++ makes it possible to design the underlying data structures on rather low level in relation to the CPU and main memory. There are several benefits in this effort:

- **Effectivity:** The structures are designed for the subject of this work. They are used to access and to manipulate large sets of the atoms.
- **Safeness of programming:** The structures provide automated management of dynamically allocated memory.
- **Debugging:** The boundaries and other conditions of the structures are checked for each operation. The debug mode can be optionally disabled for release or performance test versions.

The basic structures include the templates for 3D vector, general vector of varying length, matrix  $3 \times N$ , matrix  $4 \times 4$  and automatic pointer (smart pointer).

The smart pointer counts the number of references to the allocated block of memory. The memory is automatically released when the number of references reach zero. The main requirement for this technique is availability of the destructors in programming language since the number of references is decreased with expiration of the object. This approach is preferred to garbage collecting from the aspect of best performance.

The vectors of varying lengths combine classic arrays and dynamically linked lists. They offer random access to the elements and fast addition of new elements to the end. The allocation of memory is based on smart pointers and allows shallow copies with sharing of the data. The amount of memory can be preallocated in order to use the vectors in dynamic mode. The vectors can be mapped to other vectors and to a column or a row of a matrix to share the same elements. A vector can be either compact or sparse. Sparse vectors are used for mapping to the columns of the matrices.

The matrices 3 times n store n rows in three columns. This structure is optimized to store the coordinates of the atoms and it is used both for data processing and visualization of the atoms. The storage of the elements is row-oriented. The allocation of memory is based on smart pointers and allows shallow copies with sharing of the data. The amount of memory can be preallocated.

4x4 matrices are designed for transformations with 3D coordinates. Their content can be passed to the OpenGL using *glLoadMatrix* function.

## 5.2 Atomic structure

The basic structures are used to implement advanced representations of the data. Effective representation of the atomic structures is the most important.

The atomic structure can be loaded from PDB (Protein DataBank) and XYZ files or created from scratch. The atomic structure is composed from the atoms and the bonds. The input files usually contain only the atoms and the bonds need to be re-computed. It is also necessary to update the bonds during the interactions with the atoms when new connections between the atoms are created and others can be removed.

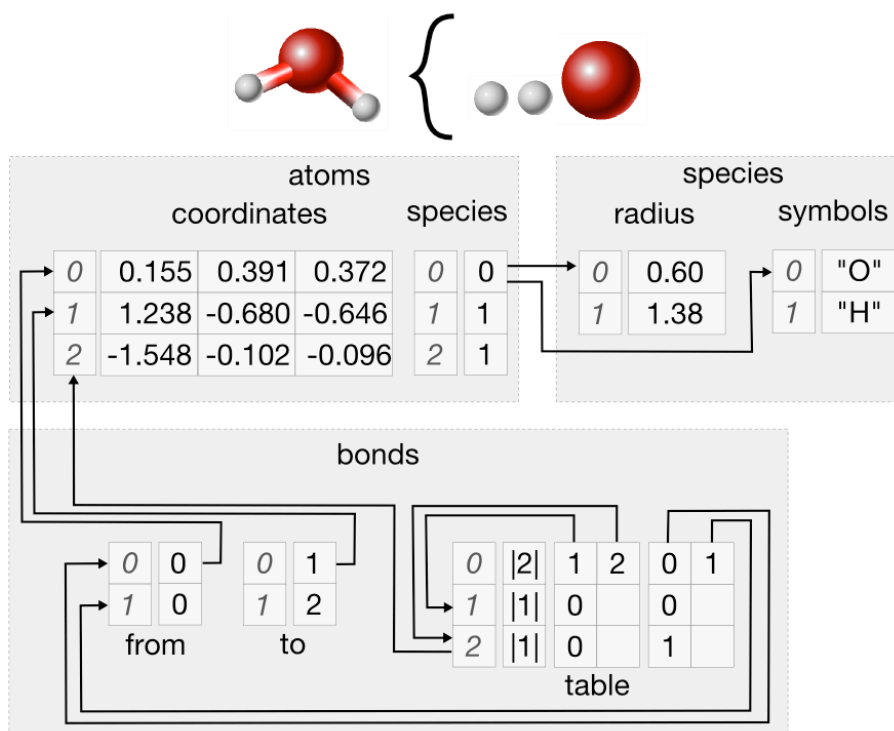


Figure 5.1: Memory representation of a water molecule can be divided to three sections: atoms, species and bonds. The sections are connected with indexing. The atoms are represented with the coordinates in Bohrs and the species of the atoms. The species contain the information about a radius, chemical symbols and the colors of the elements. The bonds store the connections between the atoms in two parts. The first part allows to access the atoms based on a bond (from-to). The second part (table) allows the access to the bonds based on the atoms.

The atoms are represented in a matrix with three columns according to three dimensional coordinates (*Fig. 5.1*). The bonds are represented in two structures. The first structure contains two arrays of indices used to index the atoms. Each pair of indices between the arrays represents one bond. The second structure organize the bonds to corresponding atoms. Using these two representations it is possible to find the atoms from the bonds and to find the bonds from the atoms.

The algorithm used to find new bonds has to be fast since the bonds are updated in real-time during user interactions. There are two main optimizations involved: spatial structures for the atoms and special thread used to find the bonds.

The atoms are further classified to two parts. The first part contains all unselected atoms and the second contains all selected atoms. Each part has its own grid (*Fig. 5.2*) with spatial partitioning of the atoms. This division brings additional acceleration since new bonds are created only between selected and unselected atoms during the interaction.

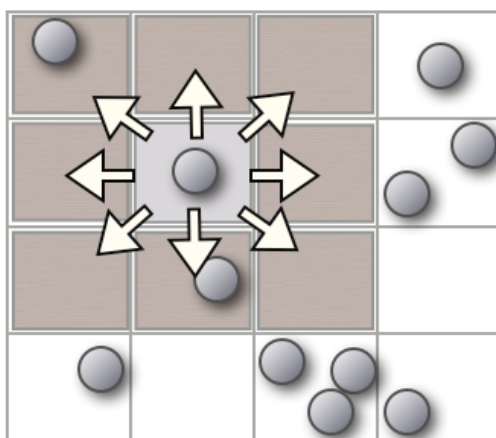


Figure 5.2: The grid shown in two dimensions where each cell can have up to eight neighboring cells. The number of neighbors in three dimensional grids is 26. The grid classifies the atoms to individual cells. The dimensions of the cells are determined by the maximal length of the bonds. Only the atoms from neighboring cells can be connected. The grids are represented in linear arrays as a sequence of cells. All elements of a single cell create an interval with two indices to the array.

The bond searching algorithm can be also running in a separated thread in the background so there are no delays in the interaction. The bonds are updated soon as the thread finishes the task. The thread can be canceled at few cancelation points if there are new coordinates of the atoms and the thread is still computing with the old coordinates.

The multiplicity of the bonds can provide further inside to the atomic structures. Multiple bonds are determined automatically as a part of bond searching algorithm. Single and multiple covalent bonds are formed from shared electrons between the atoms as a way to achieve stable electron states [11]. The algorithm is based on the number of electrons in outer atomic orbitals (shells) and creation of shared electrons between the atoms. The number of shared electrons is visualized with multiple bonds. The results are not exact, because other types of bonds and chemical processes exist which are not covered in the computation.



## 5.3 Visualization

Using a computer to visualize and interact with the data it is essential to create an application window in underlying operating system. GLUT toolkit is used for management of the windows and standard input devices. GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs [20].

The basic structures in visualization (*Fig. 5.3*) are the views, the windows and the scene graph. The scene graph represents the data in the scene.

The application has one window by default or can be opened in multiple-window mode. Each window has its own OpenGL context. A window shows one up to four views to the scene. All viewports from one window need to be re-rendered after one of them is modified. This time-consuming step is not necessary with multiple windows where each window can be dedicated to a different view.

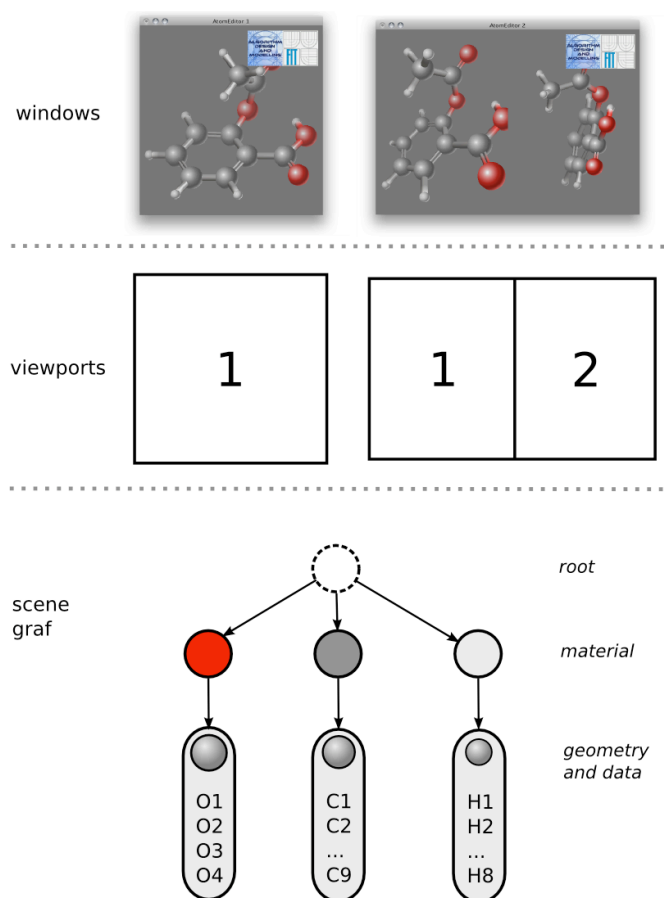


Figure 5.3: The basic structures in visualization can be seen on the image. The application can be opened in multiple-window mode (top). Each window can have up to four viewports (middle). All viewports are rendering the same scene graph. The scene graph contains the atomic structure with the atoms divided to the nodes by their elements (bottom). A single material is assigned to compound group of atoms. This distribution of the atoms is effective since it will minimize the number of material switches in OpenGL [12].

### 5.3.1 Perception of 3D

The domain of visualized atomic structures is three dimensional. However the data are presented on a two-dimensional screen using OpenGL rasterized rendering and some information is lost. Perception is the process of reading non-sensed characteristics of objects from available sensory data [29].

#### Perspective projection

Perspective projection provides depth impression. The objects appear smaller the further away they are from the observer. Perspective projection is also necessary for stereo views. Orthogonal projection is still usually preferred in modeling applications since its linear transformations preserve lengths, angles and parallel lines.

#### Lighting and shading

Lighting and shading show the orientation of the objects and fog shading creates the illusion of depth. Lighting is computationally expensive even when it is done in the hardware of modern graphics cards. Pre-computed light maps stored in the textures are used to avoid processing of the lights for each vertex in the scene. The light maps also leads to low polygon representations and thus farther improving the performance.

#### Movement

The ability to rotate the camera around the scene and to see edited structure from different angles helps in visualizing the extra dimension.

#### Stereo Views

The application has two modes of rendering (*Fig. 5.4*). Quad-buffer stereo rendering is available as an extension of standard rendering to a single OpenGL buffer. The scene graph is rendered twice in stereo mode, once for each eye.

Stereo rendering creates a perception of depth [21]. Stereoscopic or binocular depth clues, in which two images of the same object, from slightly different angles, are presented separately for each eye. This is equivalent to how human mind perceive in real life [31].

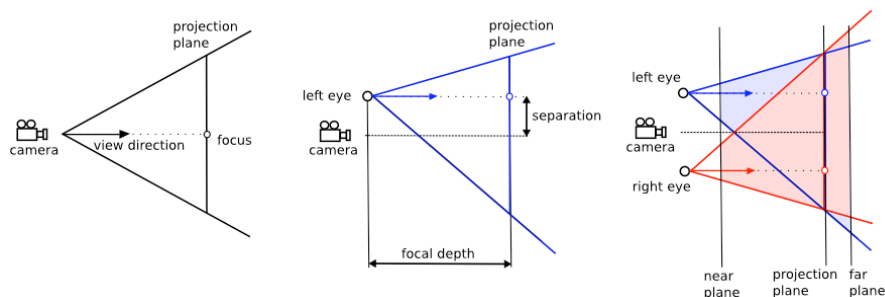


Figure 5.4: Standard rendering with one buffer (left). Stereo rendering for the left eye (middle) shows modification of camera frustum. Stereo rendering using camera frustums for both eyes is shown (right).

### 5.3.2 Scene graph

The scene graph is an object oriented approach to setup and modify the rendering pipeline during program execution. The scene graph stores all data in the scene in a collection of nodes connected to oriented graph or tree structure. One of the nodes (root) is without any predecessors.

The nodes represents elementary visualization elements including material properties, transformation matrix, geometry of the sphere and so on. Efficient interconnection of the nodes results in execution of a sequence of OpenGL commands from the nodes. The sequence is no longer encoded in a program and can be changed dynamically with node switching or addition of a new nodes.

The scene graph is traversed with a renderer in order to render one frame of the scene. The rendering pass is proceed from the top to the bottom and from the left to the right at each level of the tree structure [14]. Some nodes require multiple rendering pass.

### 5.3.3 Renderer

The renderer represents a view to the scene and contains the coordinates and the dimensions of the viewport together with camera setup. The rendering sequence is started from the renderers and then it is driven by the nodes. There are two types of renderers:

- **Hardware:** OpenGL renderer is a hardware based renderer. The functionality is provided by the nodes.
- **Software:** Postscript renderer is a software based renderer providing global rendering method with the vector output in Postscript commands.

### 5.3.4 OpenGL manager

OpenGL manager maintains all OpenGL resources in the scope of one OpenGL context. Each window has its own manager since sharing of the contexts is not possible with GLUT library. OpenGL resources includes display lists, textures, vertex buffers and vertex/fragment programs.

The manager contains a set of predefined resources to minimize memory requirements. These resources includes (among others) standard stroke font, unit sphere and unit box and they are available from the nodes.

Each class of an object that wants to allocate OpenGL resources needs to be derived from the superclass *GLConsumer*. The manager registers the consumers and assigns them the resources. A consumer can allocate resources in multiple managers.

- **pro:** Automatic deallocation of OpenGL resources.
- **con:** It is necessary to look up for the resources in the managers before they can be used in OpenGL. This is accomplished with fast binary search. The presumption of minimal loss in the performance is that the resources are used many times after each look up.

Another use of the manager is to detect and to load available OpenGL extensions. Visual quality of the outputs and maximal performance may vary between different hardware configurations. Both low-end and hi-end graphics hardware are supported.

A typical consumer is a texture or a node with some geometry.

### 5.3.5 Postscript output

GL2PS library is a standard solution to obtain screenshots in a vector format. GL2PS uses hardware rendering pipeline and OpenGL feedback buffer to piece together the projected polygons:

- GL2PS supports arbitrary geometry.
- The output quality depends on the number of polygons and high polygonal models are necessary.
- The outputs does not have jagged edges in comparison with raster images (*Fig. 5.5*).

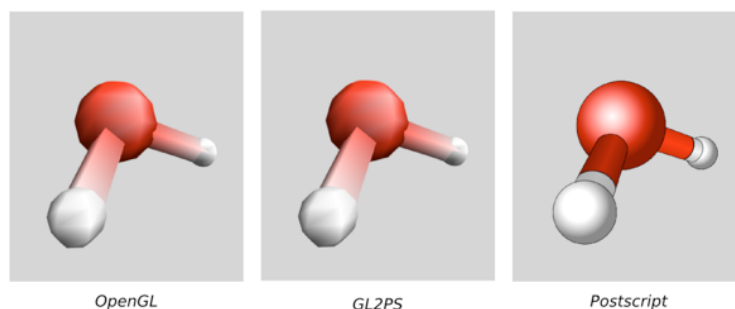


Figure 5.5: The images show a molecule of water. The output from GL2PS does not have jagged edges, but the quality is otherwise low. The result of software Postscript renderer, implemented in the scope of this work, can be seen on the right.

Software Postscript renderer implemented in this project uses the scene graph instead of polygon meshes. Higher level of description provided by the nodes allows usage of specialized Postscript commands. For example, the sphere nodes can be drawn with *arc* paths to produce rounded outlines at all zoom levels.

- It is necessary to implement planar projections for all geometry nodes (*Fig. 5.6*).
- Software rendering is slower then hardware acceleration used in GL2PS.
- The outputs have smooth outlines and shading at any scale (*Fig. 5.5*).

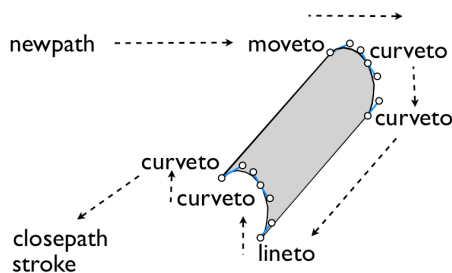


Figure 5.6: The image shows a projection of a cylinder which is used in the bond nodes. The resulting curves are drawn with Postscript commands.

### 5.3.6 Point node

The points are used as the helping coordinates to create the structures by proving the constraints. The point nodes also visualize the collisions.

Each point is rendered with billboarding which consists in textured quads oriented towards the camera. Rendering and interaction is optimized with spatial octal trees (octrees). The coordinates of the points can be used for snapping, creation of new atoms, measuring of distances and angles or as the setup coordinates for interactive elements.

### 5.3.7 Line node

The node contains a set of lines defined by their starting and ending coordinates. All lines that are stored in one node share the same thickness and pattern.

### 5.3.8 Text node

The text node contains a set of strings at custom three dimensional coordinates. All strings share the same font properties including the scale or thickness. Text nodes are used to visualize the elements of the atoms (*Fig. 5.7*) and other textual information.

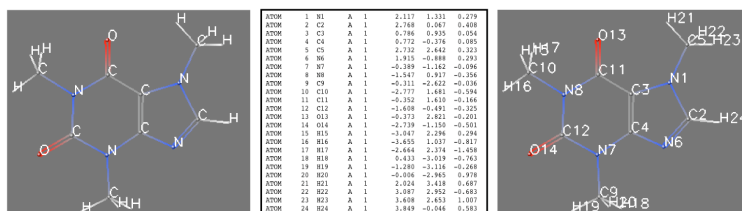


Figure 5.7: The images show a molecule of caffeine. The elements of the atoms are rendered using the text nodes (left). The order of the atoms read from the input file (center) is displayed on the right.

The strings are rendered with Hershey vector font originally created by Dr. A. V. Hershey while working at the U. S. National Bureau of Standards [22].

The font is stored in a set of display lists. Each display list represents a single character. To render a string with  $n$  characters means to render  $n$  display lists. The transformation matrix is multiplied with inverted modelview matrix in order to rotate the characters towards the camera.

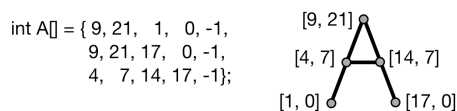


Figure 5.8: The visualization of character A is defined with three lines. The coordinates (left) and the resulting shape (right) can be seen on the image.

The characters are composed from a set of lines connected to several line strips. The lines are defined by the coordinates and the commands that start new strips similar to multiple brush strokes which are needed for drawing a character (*Fig. 5.8*).

### 5.3.9 Splat node

The splat node allows visualization of scalar fields. Scalar data can represent charge densities computed by density-functional theory (DFT) from given atomic structure. An atomic structure which was created using the editor can be displayed together with the results of simulations based on the coordinates of the atoms (*Fig. 5.9*).

Splatting, introduced by Westover [16] and improved by Crawfis [7], is a popular technique for volume rendering, where voxels are represented by Gaussian kernels, whose pre-integrated footprints are accumulated to form the image [19].

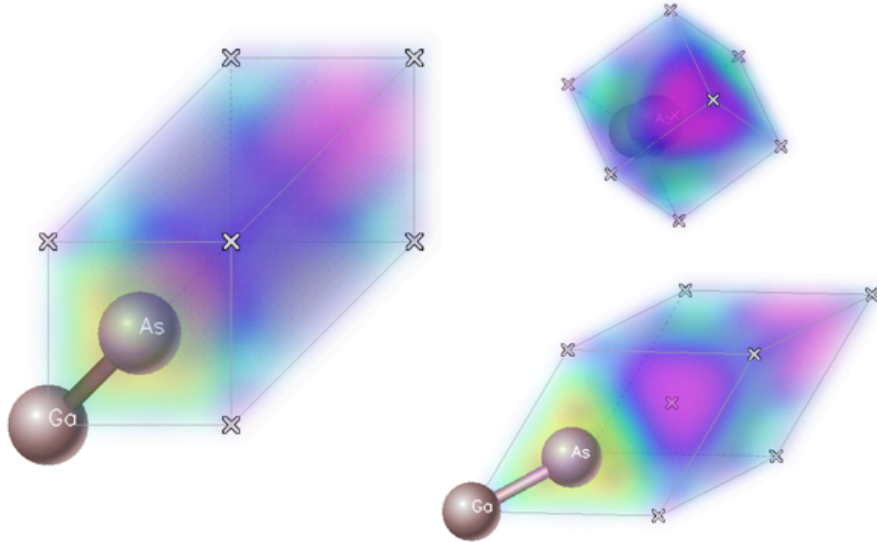


Figure 5.9: The image shows the unit cell of Gallium arsenide in three views. Both the atoms and computed densities from GaAs bulk are visualized. The density is mapped to the color spectrum. Lower densities are represented with shorter wavelengths.

An intensity integral 5.1 of electronic cloud computed by density-functional theory from given atomic structure is evaluated with splatting. The integral can be evaluated efficiently using OpenGL rendering in graphics cards rather than performing it on the CPUs.

$$I = \int_e \varrho(x, y, z) de \quad (5.1)$$

The density  $\varrho(x, y, z)$  is the probability that an electron is located at the spacial point  $(x, y, z)$  at a given time. The point is sampled on three dimensional cartesian grid. Each splat represents a single density at the point  $(x, y, z)$ .

Billboarding technique is used for rendering of the splats. A billboard is a flat object, usually a quad (square) (*Fig. 5.10*), which faces the camera. This direction usually changes constantly during runtime as the object and camera move, and the object needs to be rotated each frame to point in that direction [9].

The solution which was proposed in the Bachelor thesis is extended with hardware acceleration available in the modern graphics cards. Two additional rendering methods were implemented. The first method performs the rotation of the splats towards the camera on the graphic card. The second method uses specialized OpenGL extension for particle rendering.

## GPU Billboards

Rendering of the splats is accelerated with a vertex program. The program creates billboards in the eye space so they appear as if they were rotated towards the camera. Four points with the same coordinates are sent to the graphics card for each billboard. The coordinates represent the center of billboard polygon (*Fig. 5.10*) and they are expanded to the corners using the radius of the splats.

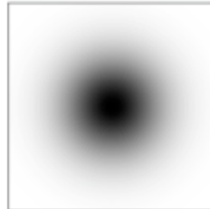


Figure 5.10: The image shows textured polygon representing one splat. The intensity of the texels is taken from Gaussian kernel.

## Point Sprites

The splats are rendered as point primitives using the Point Sprites OpenGL extension. This greatly minimize the size of transfered data to graphics hardware. The extension was originally introduced to improve particle rendering in computer games. Splatting is processed completely in hardware when using this method.

Additional vertex program is required for perspective projection. The program is used to determine the size of the splats based on the depth. The splats far away from the camera are smaller.

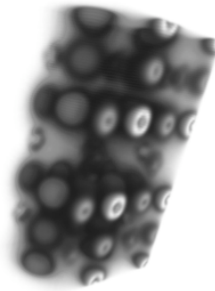


Figure 5.11: A large data set with 800000 splats can be rendered at interactive speeds.

## Buffers

The performance can be further improved with vertex and element buffers. The coordinates of all splats can be stored to a single vertex buffer and the sorting order can be stored to an element buffer. This reduces the number of OpenGL calls. The buffers usually contain large data sets (*Fig. 5.11*) and their localization to video memory using Vertex Buffer Object (VBO) and Element Buffer Object (EBO) extension is not effective.

### 5.3.10 Sphere node

The sphere node is designed to render multiple spheres with the same geometry and material properties and it is used to render all atoms of the same species. This approach is faster than when each sphere is represented by a single node.

The node supports three rendering methods (*Fig. 5.12*) depending on available hardware. The methods differ in rendering performance. The fastest method uses a combination of vertex and fragment programs. Only four vertices need to be sent to the graphics card for each sphere with this method.

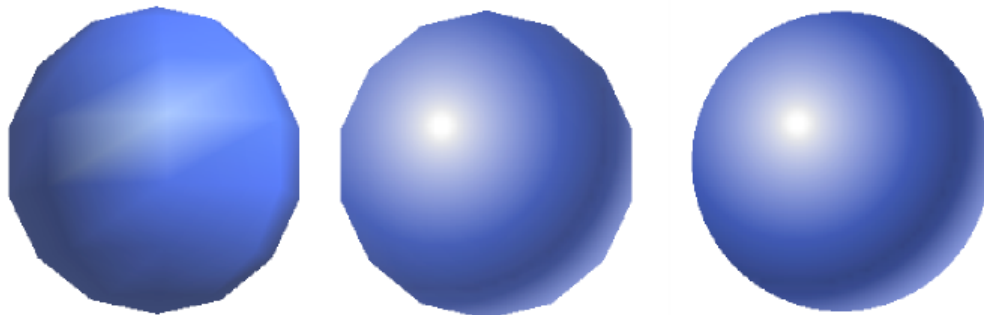


Figure 5.12: The sphere with 64 vertices in 110 triangles using standard OpenGL lighting is shown on the left. Textured sphere with pre-computed lighting and 64 vertices in 110 triangles is in the middle. GPU sphere rendered with a combination of vertex and fragment programs with 4 vertices in 1 quad on the input is in the right.

The hardware of modern graphics cards is affected by demanding requirements of computer games. There are several techniques to achieve better performance that are used in computer games. One of them are low-polygon models with higher details stored in the textures. The graphics cards are heavily optimized for texturing as the result.

The spheres are textured with pre-computed diffuse and specular maps stored in one texture to save computationally expensive hardware lighting for each vertex. The texture is mapped with Fast Phong method [10]. The number of vertices required to reach the same quality of hardware lighting is much lower. The geometry of the sphere is stored in a single triangle strip to minimize the number of processed vertices.

The other rendering method uses the shaders. Programmable shaders are the best way to reduce the number of vertices. Each sphere is rendered from four vertices linked to a single quad. The quad is rotated towards the camera inside the vertex program. The fragment program then fills a circle inside of the quad. The color of the fragments is combined from the texture and fog coordinates. The depth of the fragments is computed from their distance to the center of the quad. The resulting spheres are smoothly rounded.

The spheres are further optimized with spatial partitioning using the octal trees. The distribution of the spheres between the nodes inside the tree is based on the coordinates of the spheres. Only the spheres inside visible nodes are rendered.



### 5.3.11 Bond node

The bond node is intended for rendering of multiple bonds between two species of the atoms. All bonds that are stored in one node share the same geometry and material properties. The bonds can be visualized with multiple bond orders (*Fig. 5.13* and *Fig. 5.14*).

The geometry of the bond is represented with the cylinder created from a single triangle strip. The bonds can be also rendered as lines to reduce the number of vertices.

The bonds are colored with gradient maps stored in the textures resulting in smooth shading between two different species. The colors of the gradients are taken from the atoms on the ends.

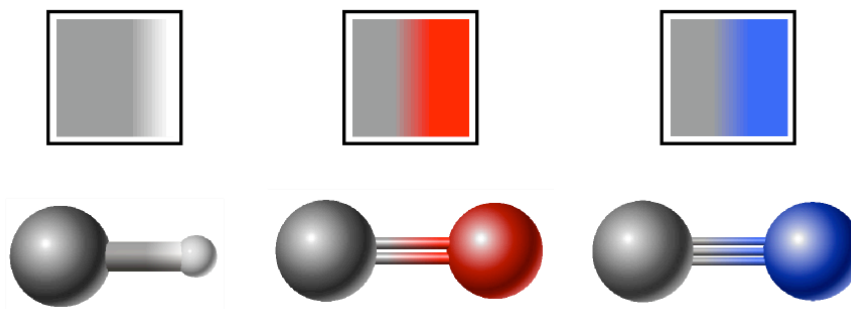


Figure 5.13: The bonds with multiple bond orders and the gradient maps. From left to right: single bond (Carbon-Hydrogen), double bond (Carbon-Oxygen) and triple bond (Carbon-Nitrogen).

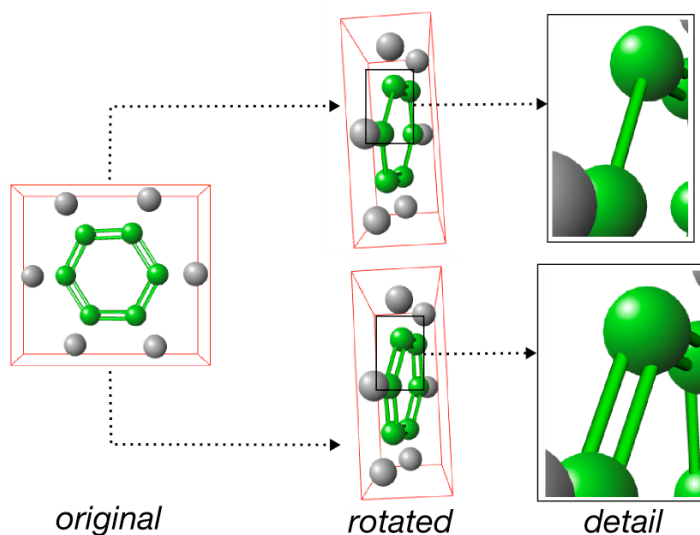


Figure 5.14: Multiple bonds are rotated towards the camera (bottom-right) to be always visible. This visualization require additional computation based on view direction.

The bonds are optimized with pre-computed transformation matrices and octrees. The transformation matrices contain the translation, the rotation and the length of the bonds. Only the bonds inside visible nodes of the octree are rendered.

## 5.4 Optimization

Rendering and interaction with large number of simple elements is more difficult than when dealing with smaller sets of complex elements. The elements are grouped and processed by the groups to minimize additional computations in optimization algorithms. A spatial structure is required to quantize the elements in 3d space.

### 5.4.1 Octree

The octal tree is a three dimensional hierarchical spatial structure. The structure is created by subdivisions of the main box to sub-octants. The subdivisions continue until the new octants contain sufficient number of elements or the number of levels in the tree reach the maximum (*Fig. 5.15*). The octrees are used to accelerate rendering, selections [5.6.1](#) [5.6.3](#), picking [5.5.4](#) and snapping [5.5.5](#).

Only the elements from the nodes which are inside the camera frustum are rendered. However this method is efficient mainly for the elements with high polygon models and it can result in slowdowns for simple geometry which serves only as the input for programmable rendering pipeline. Then it is faster to render everything and utilize vertex culling in graphics hardware.

The other algorithms which are optimized with the octal trees can be viewed in two steps. The first step process the elements by quanta represented with the nodes. This step can reduce one million of tests with the elements to several thousands tests with the nodes. The second step is working only with the elements which passed through the first step.

The tree is represented in a linear array where deeper levels in the tree are stored at higher indices. The elements in the nodes are accessed through the intervals composed from two indices to the array of elements. The elements are permuted according to the order of the nodes.

The octal trees are present in the points, the spheres and the bond nodes in a form of optimization task running in a thread. Each node has its own octree and one thread. The optimization task is started for a new set of coordinates after the same data are used for several times so there are no unnecessary optimizations during animations or when the new coordinates in the node are used just once.

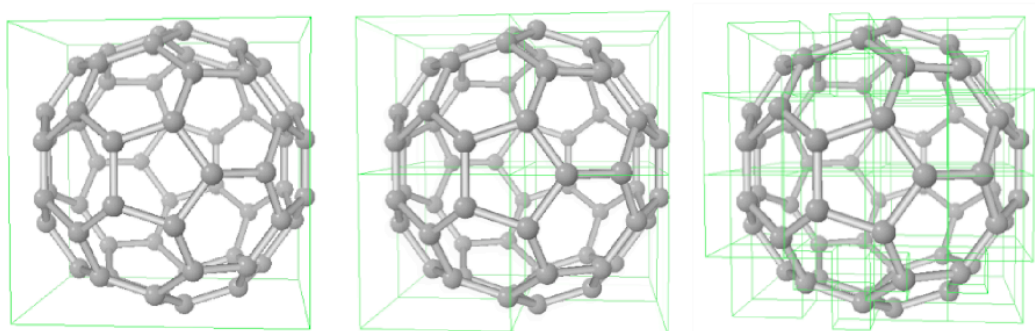


Figure 5.15: The octrees created for a C60 fullerene can be seen on the images. The number of levels in the trees is growing from left to right. The nodes are visualized with green lines.

### 5.4.2 Level of detail

Level of detail is a technique to reduce the number of details in the scene. The objects far away from the camera are rendered with simplified geometry minimizing the number of processed vertices and rendered triangles. The current level of details is determined from the distance to the camera (*Fig. 5.16*). The elements in the scene are processed in quanta to determine their level of detail. The quanta are taken from the nodes in octal trees. All elements from the same node have always the same level of detail.

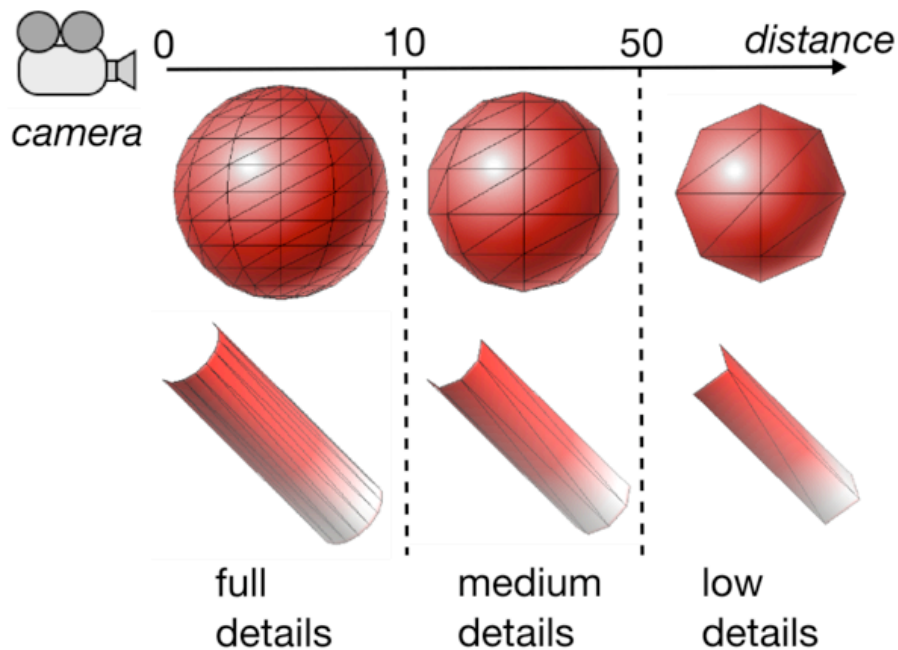


Figure 5.16: Polygon meshes used for the atoms and the bonds with three levels of details. The actual number of the meshes is higher in order to minimize visible transitions between consequent representations. The number of polygons in the meshes with medium details is determined with linear interpolation between low and full detailed meshes. The thresholds along the distance axis are exponential function.

There are several levels of details in the sphere and the bond nodes where each level is represented with a different geometry.

Level of detail is not applied for rendering methods which are based on programable pipeline since their geometry is already reduced to a single quad. Moreover, the quality produced by fragment programs automatically depends on the distance to the camera through different amounts of evaluated fragments. The objects far away from the camera are smaller and thus covered by a smaller number of fragments. One may consider use of multiple fragment programs with different quality of the outputs as an extension in the feature versions.

## 5.5 Interaction

There are two special aspects which make this editor different from the other modeling software. The first is the number of elements in the scene and the second are automatically generated connections between them in a form of bonds.

The number of elements which are being edited can be counted in hundreds, thousands or hundred thousands as the common values. Extended tools and selection algorithms are required to maintain these amounts at satisfactory efforts and speeds.

Automatically generated bonds can be seen as non-user generated content. However it is necessary to also allow interactions with the bonds with possibilities to remove existing bonds or to create new bonds. The bonds are partially created by the user and partially created automatically. The connections between the atoms provided by the bonds are additional information used in the interactions.

### 5.5.1 Precision

Professional CAD systems are concerned with numeric precision and the coordinates are stored at double precision as the minimum in these programs. Extended doubles are not an exception. High numeric precision is required for accurate results of geometric operations based on the values. The range of used units starts with micrometers (engineering) and ends with centimeters (architecture).

The coordinates of the atoms in the editor are expressed in Bohr units. The Bohr is a unit of distance commonly used as the atomic scale unit. The Bohr radius equals about 52.918 picometers [23]. The Bohrs are also used in visualization as the coordinates of OpenGL vertices.

The values of the coordinates are represented with floats. This is suitable from the point of visualization since the graphics hardware is mostly limited to single precision and double precisions were introduced only recently [3].

The atomic structures which were created with the editor can be used as the inputs for some calculations in physics. Due to the vast physical approximations the possible high accuracy of an atomic structure editor would be lost. The physical accuracies are the bottleneck and single precision is still enough. However the feature versions of the editor should support higher precisions as well.

### 5.5.2 Transformations

The scene graph and the nodes were designed to support the interactions with the scene. Interactive transformation of selected objects is controlled by transformation nodes with 4x4 transformation matrices.

The geometry nodes connected to a transformation are visited twice during the rendering sequence. The first pass renders unselected elements in the nodes. The second pass comes through a transformation node and renders selected elements which are being manipulated. This is how the transformation matrix is applied only to selected elements. The coordinates of the elements remain untouched during the interaction. The elements in the nodes are updated with a new coordinates when the interaction is finished. As the result, the performance of the interactions does not depend on the number of transformed elements.

### 5.5.3 User experience

When learning to use a computer system, however, learners are often frightened of making errors because, as well as making them feel stupid, they think it can result in catastrophe. Hence, the anticipation of making an error and its consequences can hinder a user's interaction with a system [31].

The errors can be avoided:

- through good design with appropriate representations
- visual feedbacks
- or an undo engine

#### Feedback

The system provides several feedbacks to minimize the number of possible errors. The basic visual feedback highlights interactive elements in the scene which are lying under the cursor. The feedback is enabled only during passive motions with input devices. All buttons are released in passive motion. The user is able to detect if something will happen or not as the response for a pressed button.

The preview is more advanced form of visual feedback available during passive motions and it is suitable for complex operations where the results can be quite hard to imagine. The previews are available for copy commands which can produce thousands of new atoms.

#### Collisions

The collisions provide visual feedback about the condition of edited structures. A structure is considered as valid if there is enough of free space between the atoms. The collisions show the spots where the distance between two or more atoms is too short. The meaning of collisions can be also seen as the nuclear fusions. A structure without any collisions is valuable for upcoming calculations in physics.

#### Undo engine

Since making errors can facilitate learning, an error-free situation would seem undesirable. An Alternative strategy is to make it easier for users to detect errors once they have been made and enable them to recover relatively painlessly [31].

The editor includes unlimited undo engine with the history of all commands that were used to create or to manipulate the structure. The other commands like snapping modes, projection methods or labeling of the atoms are not stored in the history, because their results can be easily reversed. The history is linear for simpler navigation between the commands.

Each of the commands contain several parameters that are necessary in order to redo (repeat) the action. The history describes the process about how to assemble the structure. The editor can be fully parametrized in the feature versions thanks to the history.

The selections are stored in separated history which is limited to the last ten items.

### 5.5.4 Picking

Picking allows to select the objects of interest in the scene and it is essential feature of any editor. There are two approaches available about how to pick the objects:

- Bottom-up method renders all objects in the scene and decides to which object belong the pixel under the cursor. The advantage is in hardware acceleration and general solution for complex shapes. The raster is usually limited to a single pixel under the cursor for faster evaluation.
- Top-down approach is hierarchical and uses the scene graph and octal trees in the nodes to minimize the number of tested elements. The method is faster then the previous one and does not involve additional rendering. The disadvantage is that each node has to define a method for intersection with a ray which can be demanding for complex shapes.

The editor uses the top-down approach which is fast enough that it can be re-computed after each movement with the input devices. This allows extended visual feedbacks during passive motions.

### 5.5.5 Snapping

Snapping to objects allows accurate direct manipulations. If snapping is active, dragged objects snap to the nearest coordinates of visible objects in the scene including the atoms and the points. Snapping can be computed in two modes.

- Three dimensional snapping is working with 3d coordinates of the objects.
- Snapping in two dimensions is based on a planar projection of the coordinates of objects in the scene. 2d snapping provides effective interaction with 3d scene on 2d output devices, because hidden depth of the objects is ignored.

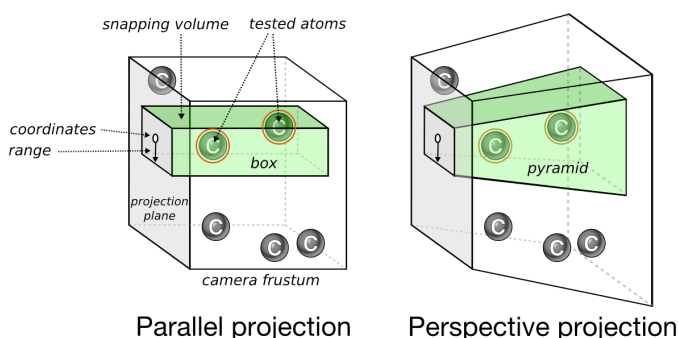


Figure 5.17: The box and the pyramid for 2d snapping.

Both modes are optimized with spatial octal trees in order to minimize the number of tested objects. The first step selects only the objects which are in the range of snapping radius. The range is represented with a sphere for 3d snapping and a box or a pyramid for 2d snapping (*Fig. 5.17*). The second step finds the coordinates with minimal distance to dragged object. Only the objects which successfully passed the first step are evaluated.

### 5.5.6 Graphic user interface

The graphic user interface is a set of components driven by the events. Since the workspace (scene) is formed by 3d space, the components and the whole interface have three dimensional character. This is yet more raised with a possibility of stereo rendering.

The components are organized to a hierarchy in a tree structure. The components include the toolbars, the docks, the tooltips and the gizmos (*Fig. 5.18*). The dock is a container with gizmos and will be described more in details later. The gizmos are interactive elements used to manipulate the scene.

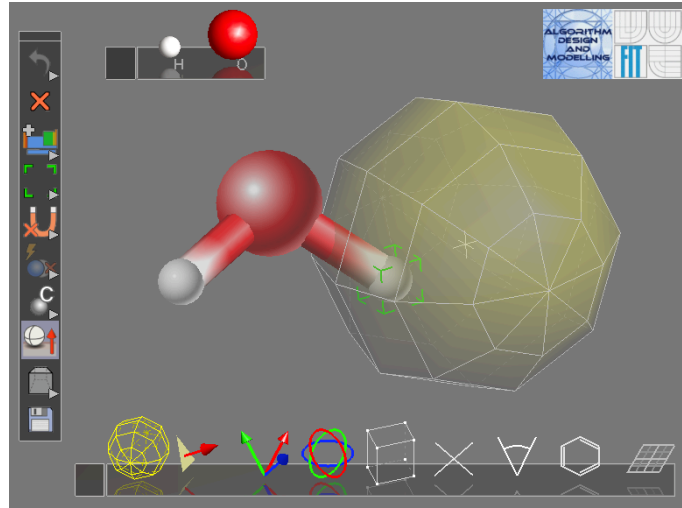


Figure 5.18: The image shows a molecule of water and graphic user interface with the main toolbar (left), the palette with the elements (top), the sphere gizmo (center) and the dock (bottom).

Some events are generated from the devices. A model device has multiple sensors. Each sensor has six degrees of freedom and a single button. All input devices are using that model. A special case is the keyboard which supplements any device in a form of modification keys like *Alt* or *Control*. The devices that provide a different type of inputs are transformed to the model device. This architecture of input devices was already proven on two exemplary devices. The first is the mouse and the second is Patriot motion tracking device with two sensors and six degrees of freedom. The events generated from the devices include button pressed, button released and motion events.

### 5.5.7 Toolbars

The main toolbar is a substitute for menu. It is faster to navigate than a menu and frequently used sub-toolbars (sub-menus) can stay opened in the workspace. All toolbars are floating. This type of user interface is well standardized across various CAD and modeling systems.

Some items in the menu have sub-toolbars with more commands related to the same group. The presence of a sub-toolbar is visualized with the arrow in the bottom-right corner of the icon. Structuring of information helps users to find the information they need.

### 5.5.8 Icons

Icons are small pictorial images that are used to represent system objects, application tools, utilities and commands. The icons can reduce the complexity of the system making it easier to learn and use [31].

There are two types of icons. The first type are two dimensional images used in the toolbars for representation of the commands. Some of these icons can be disabled (shaded) if they can not be applied. The icons can be also changed to push buttons and used as alteration switches.

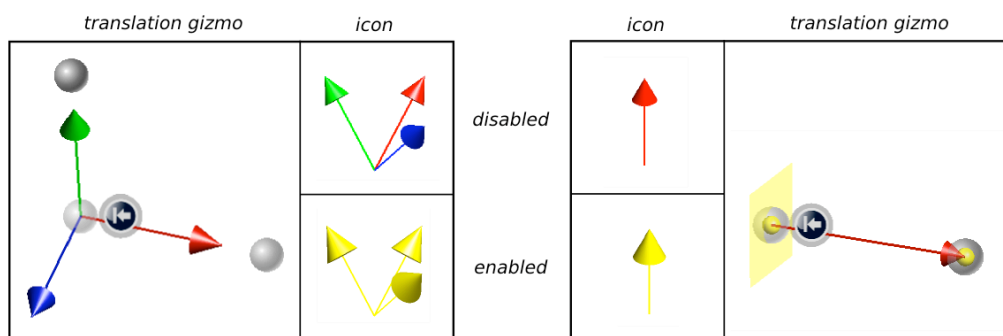


Figure 5.19: The icons reflect the current state and functionality of the gizmos. The image shows two modes of the translation gizmo.

The concept of the icons is extended with three dimensional icons. These icons are used to enable or disable interactive elements in the scene (*Fig. 5.19*).

- 3d models of the icons have better mappings with three dimensional interactive elements (gizmos) they control.
- 3d models are better fitted to 3d user interface and they are more natural with stereo rendering.
- Polygonal meshes of the icons are scalable by default. 2d icons are usually represented with multiple resolutions for this purpose.
- The models can be parametrized and updated according to the current states of the gizmos.
- Programable shaders, lighting, shadows, texturing, fog and other methods can be applied to improve visual quality of the icons.
- However 3d icons can be more difficult to design.
- 2d icons can still have better mappings that 3d icons.

The icons can display additional text information. This feature is used to show the number of available undo/redo steps, the number of the atoms in collisions or a chemical symbol of lastly used species.



### 5.5.9 Dock

The dock is a sort of toolbar except that the icons are 3d objects (*Fig. 5.20*). Each icon represents a gizmo that can be enabled or disabled. Enabled gizmos have yellow icons. The gizmos are interactive elements used to manipulate the scene. 3d icons are preferred instead of 2d icons to achieve better mapping with graphic representations of the gizmos.

Each gizmo require some user interaction in the scene. This is different from commands triggered from the toolbars.

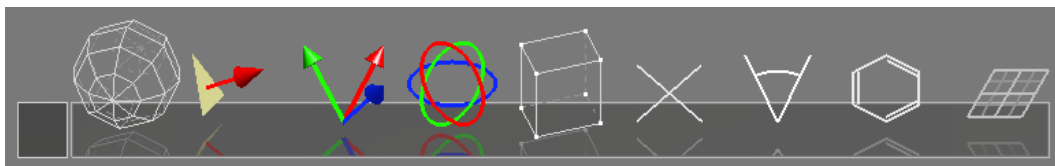


Figure 5.20: All tools for manipulation with the scene are stored in one dock.

Another usage of the docks is the palette with representations of all elements that are used in edited structure (*Fig. 5.21*). The palette allows quick modifications of the species or creation of new atoms from selected snap points.

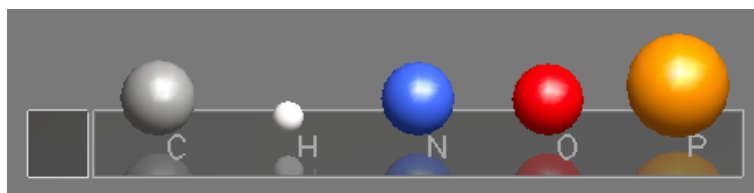


Figure 5.21: The dock contains the representations for all elements in the scene. The elements used in a DNA: Carbon, Hydrogen, Nitrogen, Oxygen and Phosphorus. 3d representations of the atoms show relative atomic radii of the atoms in the palette.

## 5.6 Gizmos

The gizmos are three dimensional interactive interactive elements for manipulations with the scene. A minimal set of standardized widgets which is shared between various applications can improve acceptance of a new software package for new users. The sphere widget from VTK is a good example of standardized widgets. However the concept of widgets is rather new and there are no predefined sets available. A new set of widgets, called gizmos, was created in the scope of this work. These gizmos are specialized for editing of the atomic structures.

Advanced gizmos can be assembled from simple interactive elements. This is accomplished with hierarchical representation of the gizmos in a tree structure. New elements can be added as the sub-nodes to extended the functionality.

Some gizmos contain a local menu with few icons. The menu can be used to customize the gizmo or to trigger specialized commands.

The functionality of the gizmos can be alternated between several modes. This is the case of translation gizmo which can operate in three axes and single axis modes.

### 5.6.1 Sphere gizmo

Sphere gizmo is designed for volumetric selections (*Fig. 5.22*). Only the objects (atoms/bonds/points) inside the sphere are selected. The sphere can be moved (left button) and scaled (right button) using the mouse. The origin of the sphere is snap-able to the atoms and to the points for easier manipulation in 3d space. The functionality of sphere gizmo is based on the sphere widget from VTK.

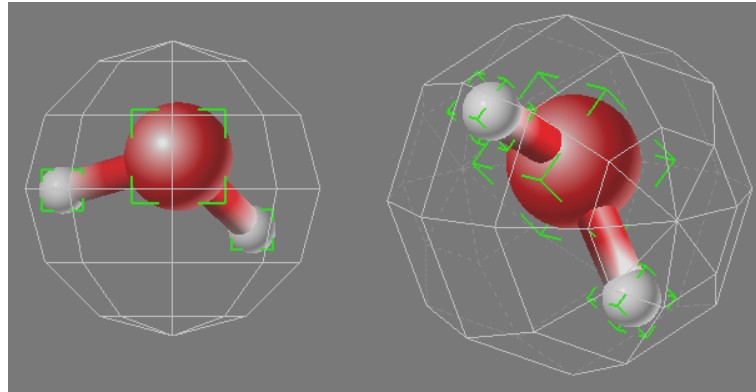


Figure 5.22: A molecule of H<sub>2</sub>O is selected with the sphere gizmo. The sphere is shown in two views.

Volumetric selections of multiple objects are optimized with octal trees. The first step selects the nodes which are intersecting with the spheres. The second step tests only the elements from these nodes.

The sphere gizmo can be used with other gizmos simultaneously. The radius of the sphere can be set with scaling gesture. The scaling gesture is based on the distance between two sensors (*Fig. 5.23*)

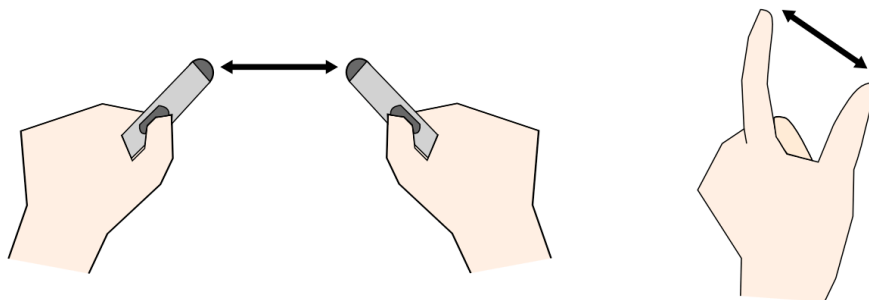


Figure 5.23: Scaling gesture uses two sensors to modify the radius of the sphere. The sensors can be represented with the devices in both hands (left) or with two fingers using a data glove (right). Scaling gesture is implemented for Patriot motion tracking device with two sensors.

### 5.6.2 Picker gizmo

The picker gizmo is a superclass for all gizmos with alternative functionality. The functionality depends on the number of picked coordinates which can be taken from the points, from the atoms or from the bonds. This way, the measure gizmo can be used to measure distances (two coordinates), angles (three coordinates) or dihedral angles (four picked coordinates). Picked coordinates are dynamic. The picker can be updated with new coordinates since it stores the identifiers of picked objects.

The picker is essential for creation of constraints. The constrains provide accurate manipulations with the scene.

All tools except the sphere and the array are derived from this class.

### 5.6.3 Plane gizmo

The planes are designed for volumetric selections (*Fig. 5.24*). Multiple planes can be created to select all objects (atoms/bonds/points) between them.

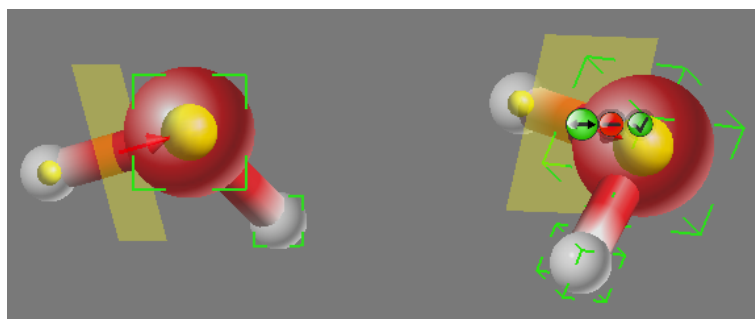


Figure 5.24: The image shows a molecule of water and selection plane in two views. Plane gizmo is used to select two atoms. The coordinates, used to create the plane, were taken from the bond between the Oxygen and Hydrogen.

The gizmo is derived from the picker superclass and therefore can be set by picking the atoms, the bonds or the points.

Volumetric selections of multiple objects are optimized with octal trees. The first step selects the nodes which are lying in the positive half-space determined by the normal vector of the plane. The second step tests only the elements from these nodes.

Each plane has a menu which allows creation of new planes or their removing. Plane normal can be flipped from the menu in order to select the opposite half-space. The menu is hidden by default. The menu is shown only when the cursor is placed over the gizmo. This eliminates the number of visible components in the interface. Structured design of the interface help users to find the information they need [31].

The plane is defined and constrained by two coordinates. The constrain is visualized with the arrow gizmo. The arrow is a basic interactive element which can be reused in more complex gizmos. The plane gizmo can be translated along the direction of the arrow to specify the selection.

### 5.6.4 Translation gizmo

The main purpose of this gizmo is to create constrained movements. The secondary function is to copy selected atoms along predefined translation (*Fig. 5.25*).

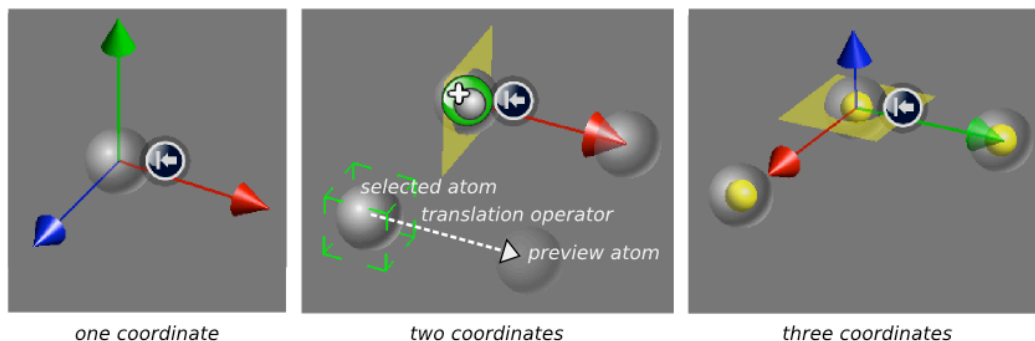


Figure 5.25: The image shows three modes of translation gizmo. The modes are set by the different number of picked coordinates. Translation operator can be created from two coordinates (middle). The translation operator has additional icon for copy command. Copy command is applied to all selected atoms and creates new atoms translated by the vector between the picked coordinates. Copy command provides a preview of the atoms that will be created.

The movements are constrained in two ways. The first is determined by the coordinates of picked objects (atoms/bonds/points). The second constrain is snapping along that direction with multipliers of  $1/4$  of it's length.

Up to three coordinates taken from the atoms/bonds/points can be picked to define the first constrain. A bond can be picked to use the coordinates of atoms on the ends.

- There is no constrain from a single picked coordinate.
- 2 picked coordinates mean translation along the line between two points.
- 3 picked coordinates mean translation in the plane defined by three points or translation along the normal of that plane.

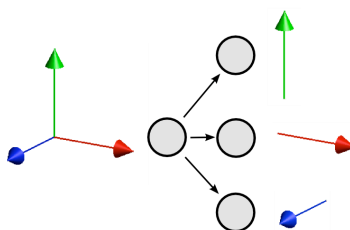


Figure 5.26: The image shows the composition of translation gizmo in the scene graph. The gizmo is build from three basic elements (arrows).

The arrows are basic build elements (*Fig. 5.26*). The arrow is a basic interactive element which can be reused in more complex gizmos. The arrows provide snapping along their direction as an additional constraint.

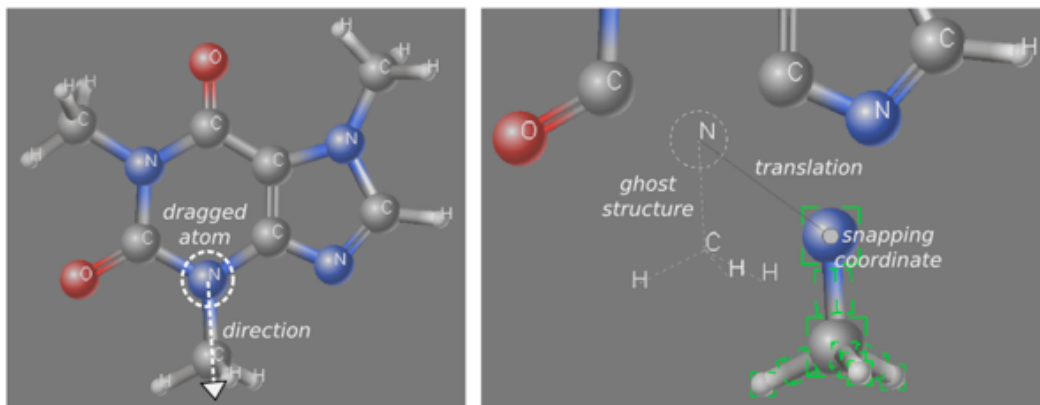


Figure 5.27: The image show two steps of direct manipulation with a molecule of caffeine. Initiation of the dragging is shown on the left and real time interaction on the right. A ghost structure with original coordinates of the atoms is displayed for reference. The group of the atoms that are being translated provides snapping coordinate so the group can be attached to some place precisely.

The atoms can be also translated just with dragging without using any gizmo (*Fig. 5.27*). Dragging is initiated at some atom. Selection of the atoms that will be translated is based on the direction of movement during initiation process. Only the atoms that are connected to the bond with minimal angle to the movement are selected.

### 5.6.5 Rotation gizmo

The gizmo is designed to rotate selected atoms (*Fig. 5.28*). The rotations are constrained in two ways. The first is determined by the coordinates of picked objects (atoms/bonds/points). The second constrain is snapping along the angle with multipliers of 30 degrees. Snapping constrains can be set precisely from the angles measured with Measure gizmo.

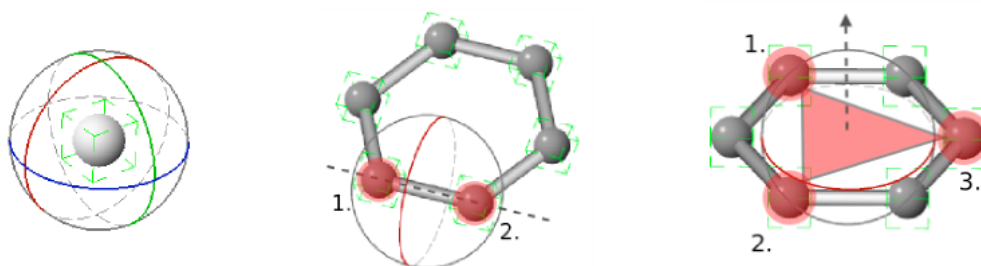


Figure 5.28: The image shows three modes of rotation gizmo. The modes are set by the different number of picked coordinates. From the left to the right: the first mode allows rotations around a pivot point, the second mode rotates the atoms around an axis and the third mode allows rotations in a plane.

Up to three coordinates taken from the atoms/bonds/points can be picked to define the first constrain. A bond can be picked to use the coordinates of atoms on the ends.

- 1 picked coordinate means rotation around that point.
- 2 picked coordinates mean rotation around the line between two points.
- 3 picked coordinates mean rotation in the plane defined by three points.

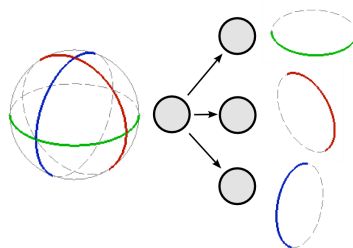


Figure 5.29: The image shows the composition of rotation gizmo in the scene graph. The gizmo is build from three basic elements (circles).

The circles are basic build elements (*Fig. 5.29*). The circle is a basic interactive element which can be reused in more complex gizmos. The circles provide snapping to the angles as an additional constraint. Snapping angles can be set from measured angles in combination with the measure gizmo.

### 5.6.6 Image map gizmo

The idea of image maps comes from HTML maps. Image maps allow authors to specify regions of an image or object and assign a specific action to each region [30].

The gizmos that are based on image maps are created from two parts: a raster image which is rendered as a texture and a map. Alternative versions of the same gizmo can be created keeping the same functionality (*Fig. 5.31*).

The maps are composed from a set of active regions. The areas of the regions are defined with the coordinates or the dimensions. Each area has a name so an appropriate action can be chosen from the application (*Fig. 5.30*).

The image map gizmos are defined in external files. The files contain a name of the background image and a set of active regions in XML format. A simple XML parser was implemented for this purpose. Short XML example taken from *periodictable.xml*:

```
<name>periodic table</name>
<image>periodictable.bmp</image>
<map>
  <area>
    <shape>rect</shape> <name>H</name>
    <x>26</x> <y>29</y> <width>41</width> <height>41</height>
  </area>
</map>
```

1 A												8 A									
H <sup>1</sup>											He <sup>2</sup>										
2 A												3 A	4 A	5 A	6 A	7 A	8 A				
Li <sup>3</sup>	Be <sup>4</sup>											Be <sup>5</sup>	C <sup>6</sup>	N <sup>7</sup>	O <sup>8</sup>	F <sup>9</sup>	Ne <sup>10</sup>				
3 A		4 B		5 B		6 B		7 B		8 B		1 B		2 B		3 A	4 A	5 A	6 A	7 A	8 A
Na <sup>11</sup>	Mg <sup>12</sup>	Ca <sup>20</sup>	Sc <sup>21</sup>	Ti <sup>22</sup>	V <sup>23</sup>	Cr <sup>24</sup>	Mn <sup>25</sup>	Fe <sup>26</sup>	Co <sup>27</sup>	Ni <sup>28</sup>	Cu <sup>29</sup>	Zn <sup>30</sup>	Ga <sup>31</sup>	Ge <sup>32</sup>	As <sup>33</sup>	Se <sup>34</sup>	Br <sup>35</sup>	Kr <sup>36</sup>			
Rb <sup>37</sup>	Sr <sup>38</sup>	Y <sup>39</sup>	Zr <sup>40</sup>	Nb <sup>41</sup>	Mo <sup>42</sup>	Tc <sup>43</sup>	Ru <sup>44</sup>	Rh <sup>45</sup>	Pd <sup>46</sup>	Ag <sup>47</sup>	Cd <sup>48</sup>	In <sup>49</sup>	Sn <sup>50</sup>	Sb <sup>51</sup>	Te <sup>52</sup>	I <sup>53</sup>	Xe <sup>54</sup>				
Cs <sup>55</sup>	Ba <sup>56</sup>	La <sup>57</sup>	Hf <sup>72</sup>	Ta <sup>73</sup>	W <sup>74</sup>	Re <sup>75</sup>	Os <sup>76</sup>	Ir <sup>77</sup>	Pt <sup>78</sup>	Au <sup>79</sup>	Hg <sup>80</sup>	Tl <sup>81</sup>	Pb <sup>82</sup>	Bi <sup>83</sup>	Po <sup>84</sup>	At <sup>85</sup>	Rn <sup>86</sup>				
Fr <sup>87</sup>	Ra <sup>88</sup>	Ac <sup>89</sup>	Rf <sup>104</sup>	Db <sup>105</sup>	Sg <sup>106</sup>	Bh <sup>107</sup>	Hs <sup>108</sup>	Mt <sup>109</sup>	Uun <sup>110</sup>	Uuu <sup>111</sup>	Uub <sup>112</sup>	Uut <sup>113</sup>	Uuq <sup>114</sup>	Uup <sup>115</sup>	Uuh <sup>116</sup>	Uus <sup>117</sup>	Uuo <sup>118</sup>				
		L	Ce <sup>58</sup>	Pr <sup>59</sup>	Nd <sup>60</sup>	Pm <sup>61</sup>	Sm <sup>62</sup>	Eu <sup>63</sup>	Gd <sup>64</sup>	Tb <sup>65</sup>	Dy <sup>66</sup>	Ho <sup>67</sup>	Er <sup>68</sup>	Tm <sup>69</sup>	Yb <sup>70</sup>	Lu <sup>71</sup>					
		A	Tn <sup>80</sup>	Pa <sup>91</sup>	U <sup>92</sup>	Np <sup>93</sup>	Pu <sup>94</sup>	Am <sup>95</sup>	Cm <sup>96</sup>	Bk <sup>97</sup>	Cf <sup>98</sup>	Es <sup>99</sup>	Fm <sup>100</sup>	Md <sup>101</sup>	No <sup>102</sup>	Lr <sup>103</sup>					

Figure 5.30: The image shows the periodic table of elements as an example of image map gizmos. The regions and their names correspond to the elements. The periodic table is used to change the species of selected atoms or to create new atoms at selected points. New species are added to the palette with all elements that are used in the structure. The table can be used just to populate the palette.

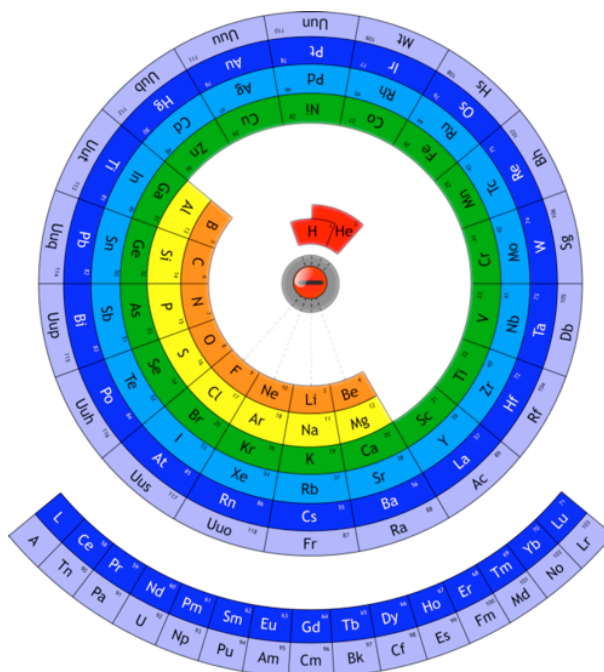


Figure 5.31: The image shows the alternative version of periodic table gizmo. The alternative versions of the same gizmo can be created easily by using a different background image and providing a new set of active regions. The circular form of the periodic table was designed by Mohammed Abubakr.

### 5.6.7 Array gizmo

The array gizmo is a form of simplified crystal builder. To the extent that the gizmo is basically a small application. The array allows to setup and to repeat the unit cell (*Fig. 5.32*).

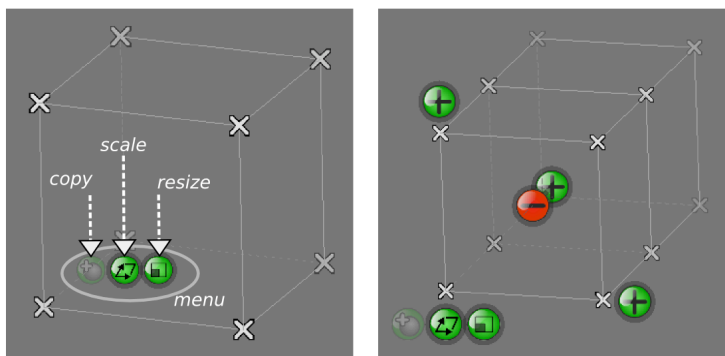


Figure 5.32: The image shows the array gizmo. The unit cell is repeated for two times along one axis on the right part of the image. The plus and minus symbols control the number of repetitions along the axis they lie on. Resizing can be enabled from the local menu of the gizmo. The other items in the menu allow to setup the unit cell (scale) and to copy selected atoms to each repeated cell in the array (copy).

The array also provides several snap points. The points are automatically created in all corners of the cells. New important points in a symmetry of the crystals can be based on these points. The array can be used with other gizmos simultaneously.

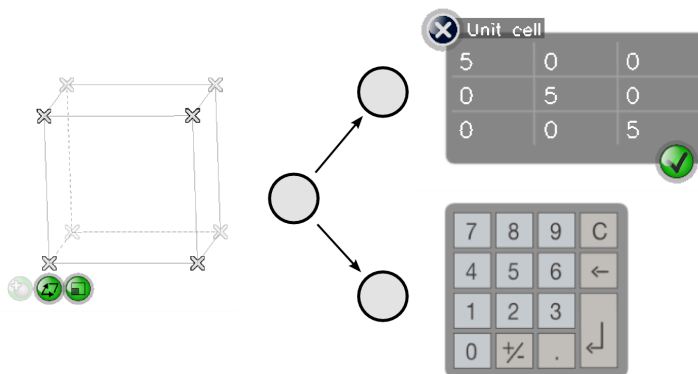


Figure 5.33: The image shows the composition of array gizmo in the scene graph. The gizmo is extended with the matrix editor and the numeric keyboard. The matrix editor is used to setup the values in 3x3 matrix. The numeric keyboard can be convenient for a touch screen input devices. The keyboard is connected to the matrix editor. A real keyboard can be used as well.



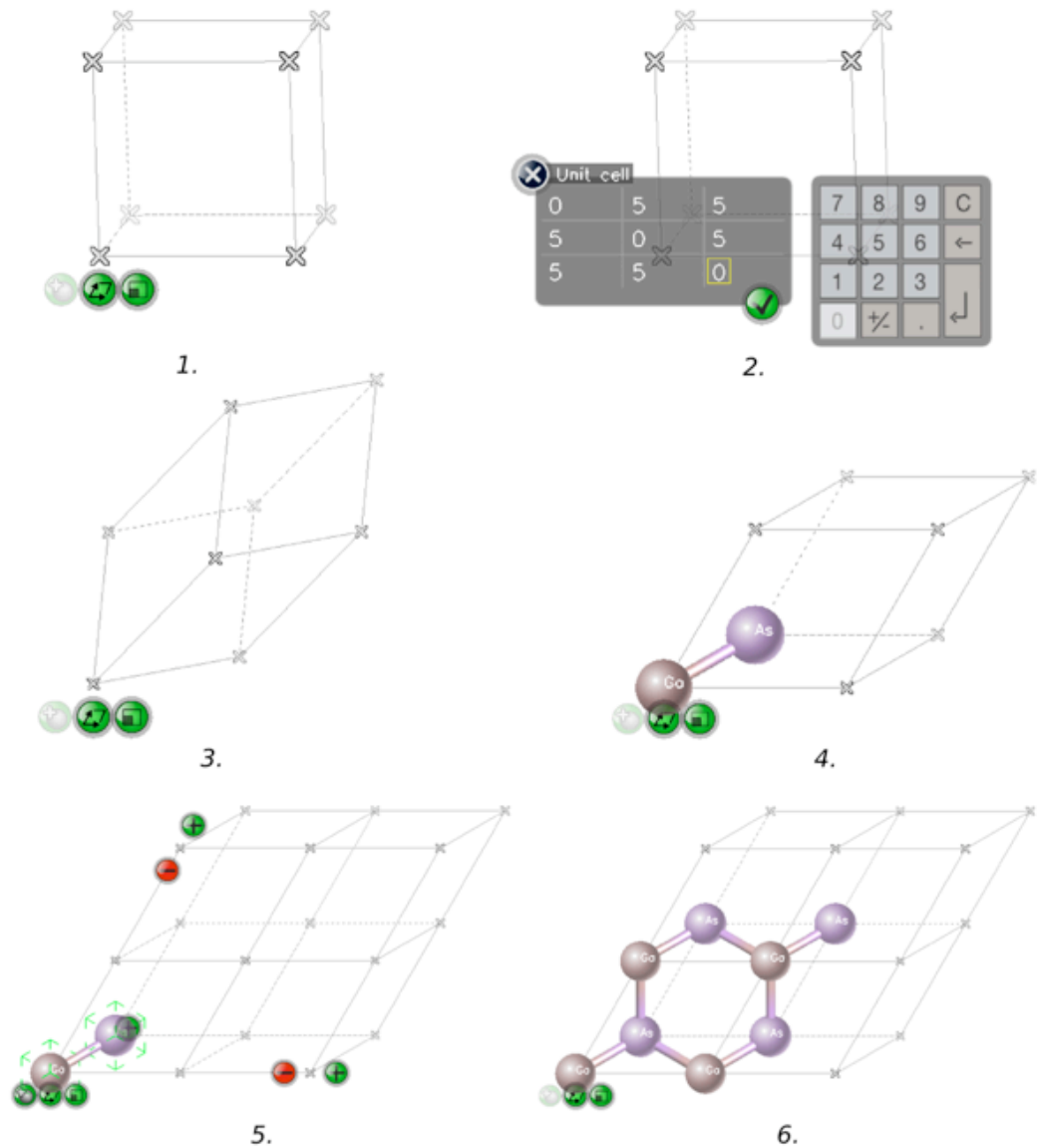


Figure 5.34: A crystal of Gallium arsenide is built in four steps displayed in six sub-images including the inter-results. The original unit cell (1.) is modified with the matrix editor (2.) to the unit cell of GaAs (3.). The atoms are created using the diagonal of the cell (4.). The array is resized to 2 times 2 grid (5.) and selected atoms from the first cell are copied into each repeated cell (6.).

### 5.6.8 Snap point gizmo

New snap points can be created with snap point gizmo (*Fig. 5.35*). The snap points provide helping coordinates for accurate manipulations with the atoms. New atoms can be instantiated at selected points.

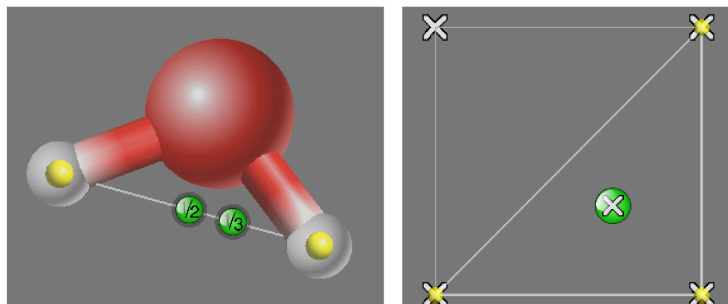


Figure 5.35: Snap point gizmo is defined between two hydrogens (left). Three points from the array gizmo were picked to define the gizmo (right).

Up to three coordinates taken from the atoms/bonds/points can be picked to define the gizmo. A bond can be picked to use the coordinates of atoms on the ends.

- 2 picked coordinates allows to create new point in one half or one third of the distance between the coordinates.
- 3 picked coordinates allows to create new point in the center of the coordinates.

### 5.6.9 Measure gizmo

The measure gizmo can be used to measure distances, angles (*Fig. 5.36*) and dihedral angles (*Fig. 5.37*) between picked objects. It is possible to create multiple measurements. The measured values are dynamic, which means that they are automatically updated as the picked objects change their coordinates.



Figure 5.36: A molecule of water with measured angle can be seen on the left. The right part of the image shows the local menu of the gizmo. The menu can be accessed by clicking on the measured value and closed with the x symbol. The menu allows remove the measured value (minus) or to create additional measurements (check).

Up to four coordinates taken from the atoms/bonds/points can be picked to define the gizmo. A bond can be picked to use the coordinates of atoms on the ends.

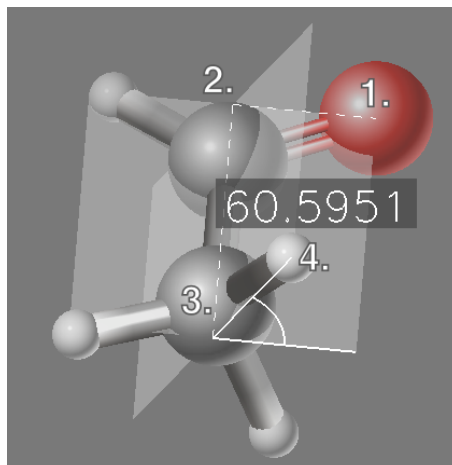


Figure 5.37: The image shows a molecule of acetaldehyde with measured dihedral angle.

- A distance is measured between two picked points.
- An angle is measured between three picked points.
- A dihedral angle is measured between four picked points (*Fig. 5.37*).

#### 5.6.10 Bonds gizmo

Bonds gizmo can create new bonds and remove or change existing bonds. The gizmo can be useful for special cases which are not covered with automatically generated bonds. Multiplicity of the bonds can be alternated to fix incorrect bond orders which were generated automatically.

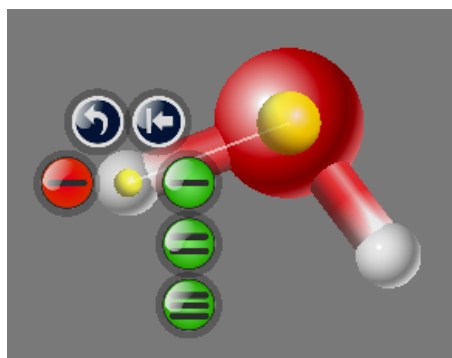


Figure 5.38: The image show a molecule of water and the bonds gizmo. The gizmo is defined between the oxygen and left hydrogen atoms. The gizmo has a local menu. The menu has circular organization of the icons. The bond can be removed (red minus) or changed to a different bond order (two lines or three lines).

At least two coordinates need to be picked in order to open the menu (*Fig. 5.38*). Multiple coordinates can be picked to create a path. The actions which are chosen from the menu are then applied to all pairs of the atoms along the path.

### 5.6.11 Construction plane gizmo

The main functionality of this gizmo is to setup the camera precisely. The secondary functionality is to define a construction plane.

The construction plane can be created only from three picked coordinates. A preview of the grid is shown for verification of correctly picked coordinates. The grid provides snapping constraints for the atoms. The construction plane can be used with other gizmos simultaneously since it remains visible even when the gizmo is disabled.

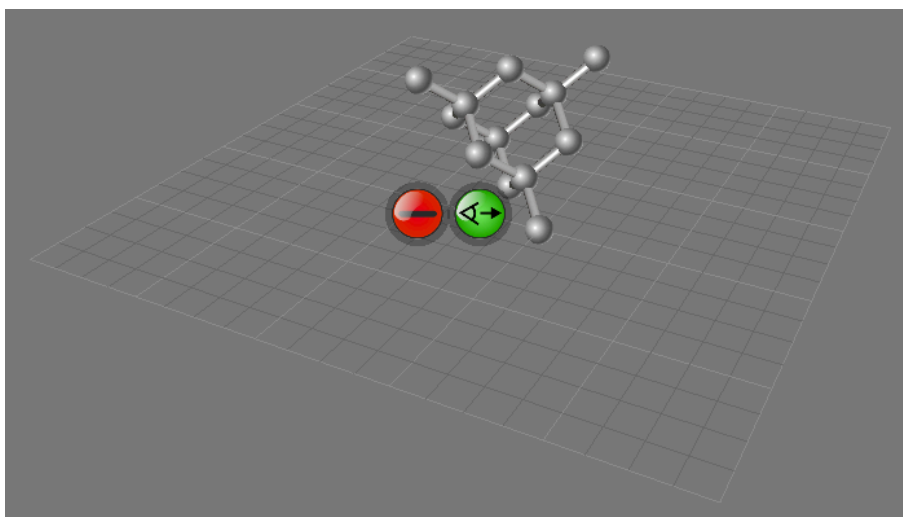


Figure 5.39: Construction plane is placed below a structure of diamond. The gizmo has a local menu. Two items of the menu are visible on the image. The first item (minus) removes the plane. The second item (look at) rotates the camera in a way that the construction plane becomes parallel to the projection plane. The rotation is animated into several steps, because sudden changes with the camera can cause disorientation.

- One picked coordinate can be used to shift the camera so that point becomes in the center of the viewport.
- Two picked coordinates can be used to rotate the camera in the current projection plane.
- Three picked coordinates can be used to rotate the position of the camera. This usually results to a different projection plane. New projection plane is defined by three picked coordinates.

# Chapter 6

## Experiments

The chapter describes several tests that were performed to choose the best solution of the problems. The second part contains measured results of selected test scenarios to validate the decisions. The most of the tests are oriented to visualization. This is caused by demands for high performance of rendering pipeline which results in short responses during the interaction. The interaction with the atoms is impossible without a decent number of rendered frames per second.

### 6.1 Setup

In the beginning it was important to choose the most suitable application framework and to explore available visualization toolkits. The framework should be system independent, simple and with possibility for writing OpenGL programs.

#### 6.1.1 Qt, VTK and GLUT

The first version was implemented with visualization toolkit VTK [13] and multi-platform application framework Qt [26]. Simple transformation widget was created as a test case interactive element. The resulting source code of the widget contained over one thousand of lines. The most of those lines belonged to VTK calls and only few lines described the actual functionality. It is also difficult to apply hierarchical approach and assemble the widgets from basic interactive elements since organization of the actors is linear in VTK. The interaction events from input devices are fixed to two dimensional mouse coordinates and therefore inappropriate for extensibility with additional input devices.

GLUT was found to be adequate system independent solution for the implementation. The simplicity and lack of ready to use user interface invites to experiment.

#### 6.1.2 Structures

The reason for this test was to establish the best memory representation for a set of three dimensional coordinates. The array of 3d vectors (structure of structures, heterogenous) was compared with the matrix stored in linear array (a single homogenous structure). The sequence of instructions and measured times are better for the linear representation.

### 6.1.3 Input devices

A simple OpenGL application was developed to figure out how to work with Patriot motion tracking device. The coordinates of the sensors are used to draw freehand lines in 3d space (*Fig. 6.1*). The thickness of the lines depends on motion speed.

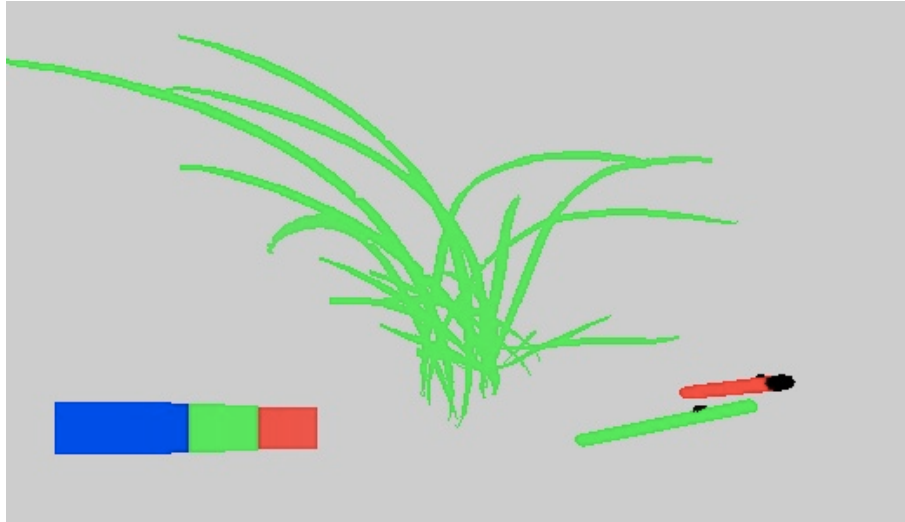


Figure 6.1: The image shows several drawn lines, three color cubes to change brush color and two avatars of the sensors for easier orientation.

The drivers for Mac OS X and device listener were implemented as the result of this test. The driver was reverse engineered by monitoring of USB packets from Windows driver. The device listener provides the interface to system dependent communication with a driver.

The listener contains a thread which observes the coordinates and other attributes provided by the driver. The values are transformed to the events which are synchronized with the application using the timers.

## 6.2 Visualization

Several tests were performed to measure the performance of rendering pipeline. The basic performance criterion in all tests is interactivity. The human brain can distinguish individual frames up to around 24 frames per second (fps). However even lower values like 15 fps can be still perceived as a smooth motion.

The tests are running in default 400 x 400 window, if not specified otherwise.

### 6.2.1 Splats

The test performs rendering of large volumes of the data using different rendering methods. The results are shown in the graph (*Fig. 6.2*).

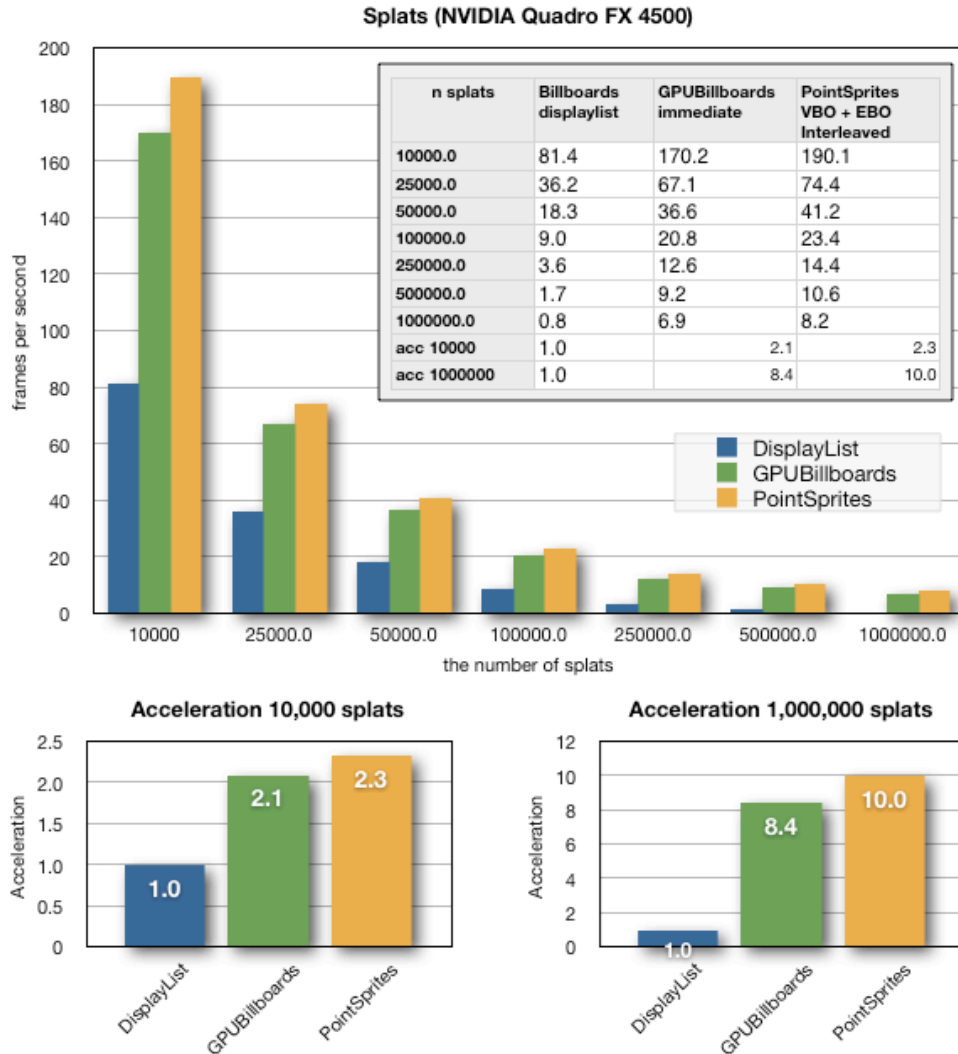


Figure 6.2: The graph on the top shows the number of rendered frames per second depending on the number of splats and used rendering method. The splats are generated in a cube at random coordinates. The number of splats goes from ten thousands to one million. The acceleration is more effective with higher numbers of the splats (two sub-graphs on the bottom). Ten times more splats can be rendered with the PointSprites method (1 million splats) in comparison with display lists (100 thousands) at the same frame rates. Visualization of 1 million splats is still interactive with PointSprites rendering method (8.2 FPS).

## 6.2.2 Spheres

The spheres represent the atoms. High performance is necessary in order to render large molecules. The results are shown in the graph (*Fig. 6.3*).

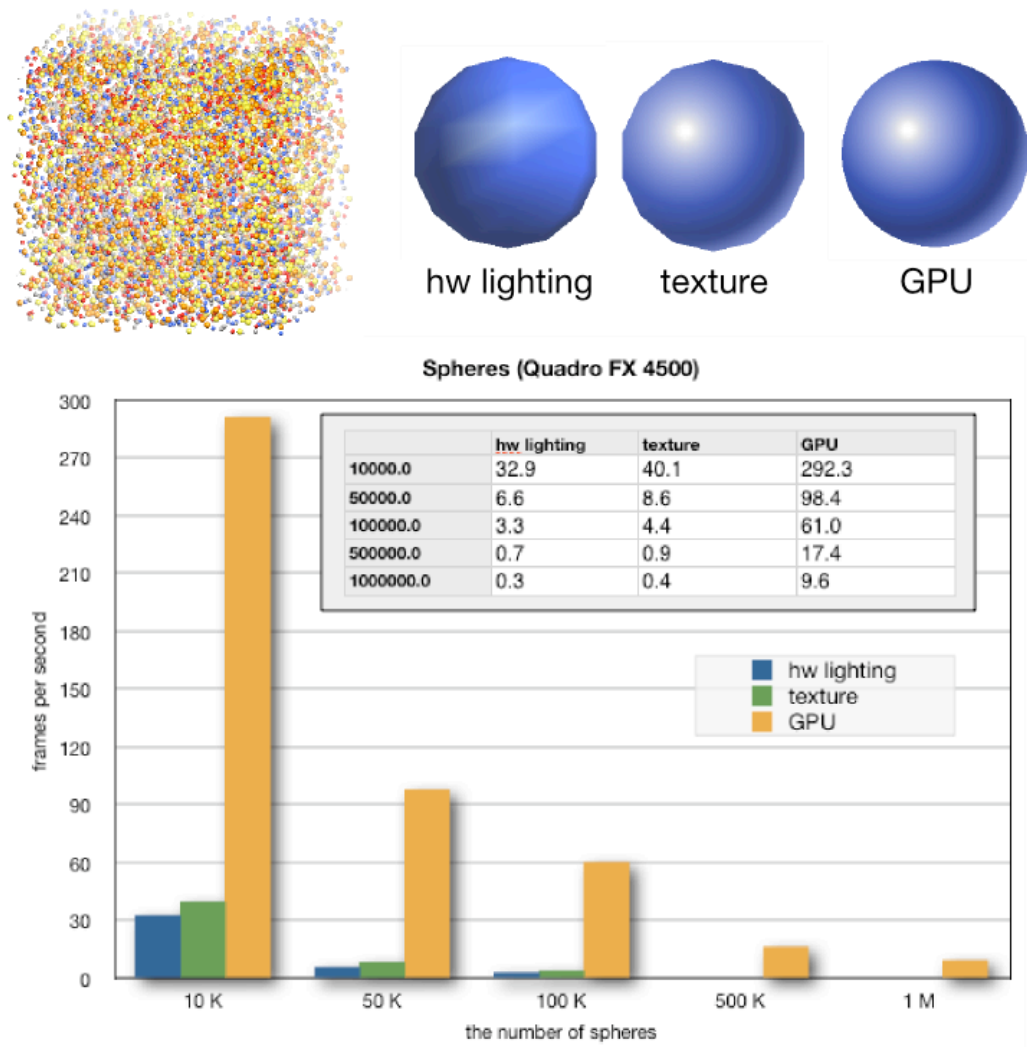


Figure 6.3: The graph shows the number of rendered frames per second depending on used rendering method. The number of the spheres goes from ten thousands to one million. The atoms, represented with the spheres, are generated in a cube (top-left corner) at random coordinates and divided between five species to involve switching of the textures and materials. The spheres rendered with the shaders on a GPU are the most effective. One million of GPU spheres is still interactive with almost ten frames per second. That gives 104 ns to render one sphere as the best result. The other two rendering methods are less effective. The textured spheres are a little bit faster than the spheres rendered with hardware lighting. This means that the cost of texturing is preferable to hardware lighting even that T&L (Transform and Lighting) units are boosted up in Quadro graphics cards.



The molecules are rendered also with the bonds. This test examines the efficiency on real samples (*Fig. 6.4*).

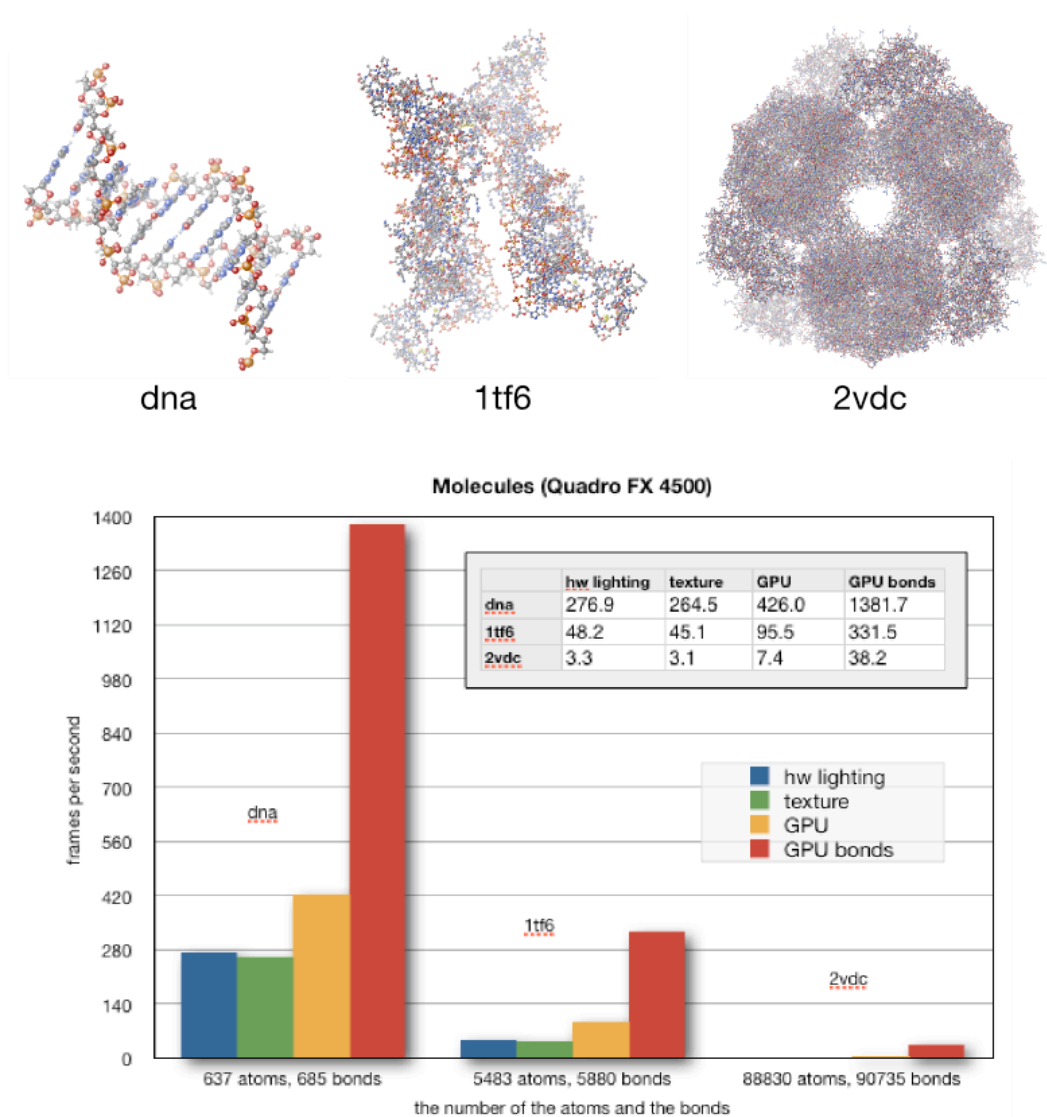


Figure 6.4: Three molecules (top) are tested with different rendering methods. The spheres are complemented with the bonds to evoke real conditions. The molecules differ in the number of atoms and bonds. The largest molecule contains almost 90 thousands of atoms. GPU bonds is an additional rendering method for the bond nodes. Each bond is rendered only with four vertices which are processed by vertex and fragment programs in this method. However rendering of the bonds on a GPU was not solved completely and remains experimental. The combination of GPU spheres and GPU bonds, when the whole atomic structure is rendered with the shaders, has the best results.

The rendering pipeline can be divided to two steps. The first stage process the geometry and the second fills the buffer. The load of these parts depends on used rendering method (*Fig. 6.5*).

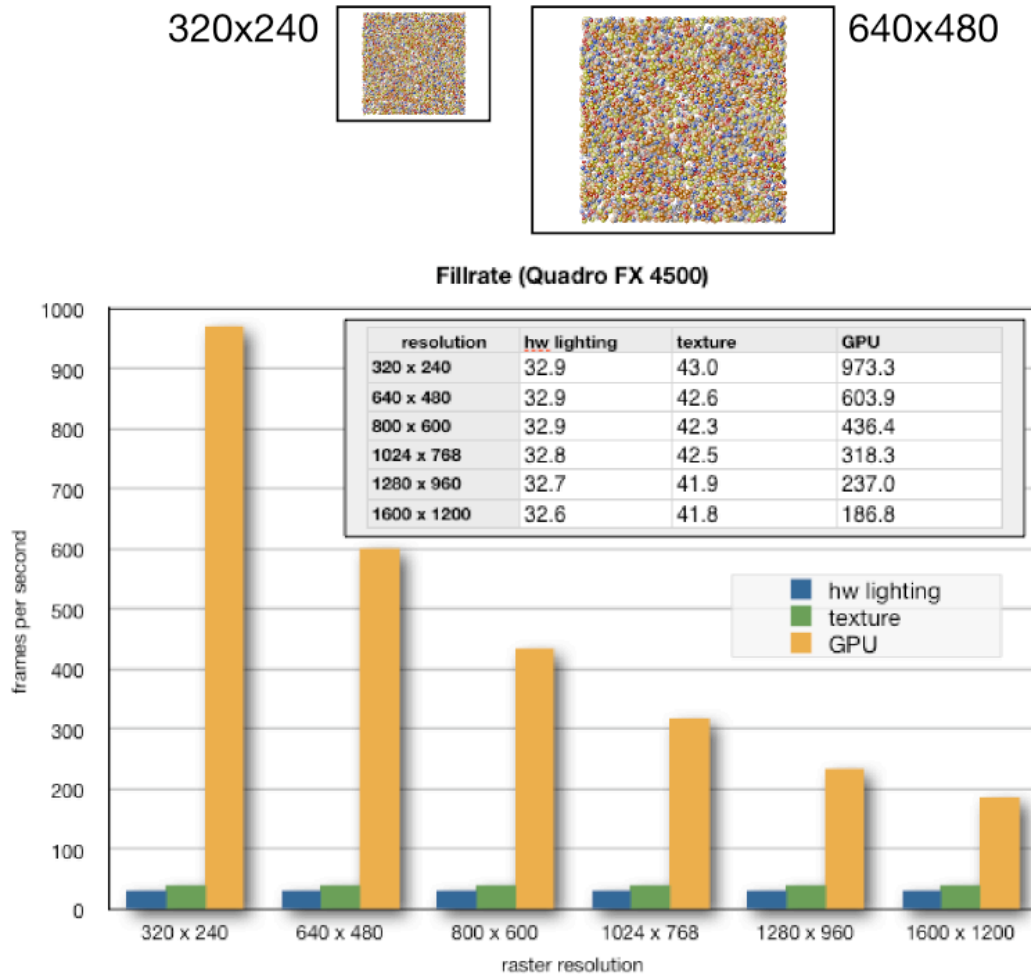


Figure 6.5: This test is the result from determination of the bottlenecks. The same number of spheres (ten thousands) is rasterized with various resolutions of the buffer. It can be seen that the first two rendering methods with complex polygonal meshes have almost constant framerates across all resolutions. This is not necessary good since it indicates that the rendering pipeline is probably blocked. The third method shows completely different behavior where the performance heavily depends on the resolution. The spheres are rendered with programable rendering pipeline in this method. The fragment program has fourteen instructions that compute the shape of the circle, the depth and resulting color from a texture and fog coordinates. The program is executed for most of the fragments in the test (top).

### 6.2.3 Materials

OpenGL acts as a state machine. Each modification of the states can lead to additional time consuming reconfiguration. That is why lower numbers of state changes often lead to higher framerates (*Fig. 6.6*).

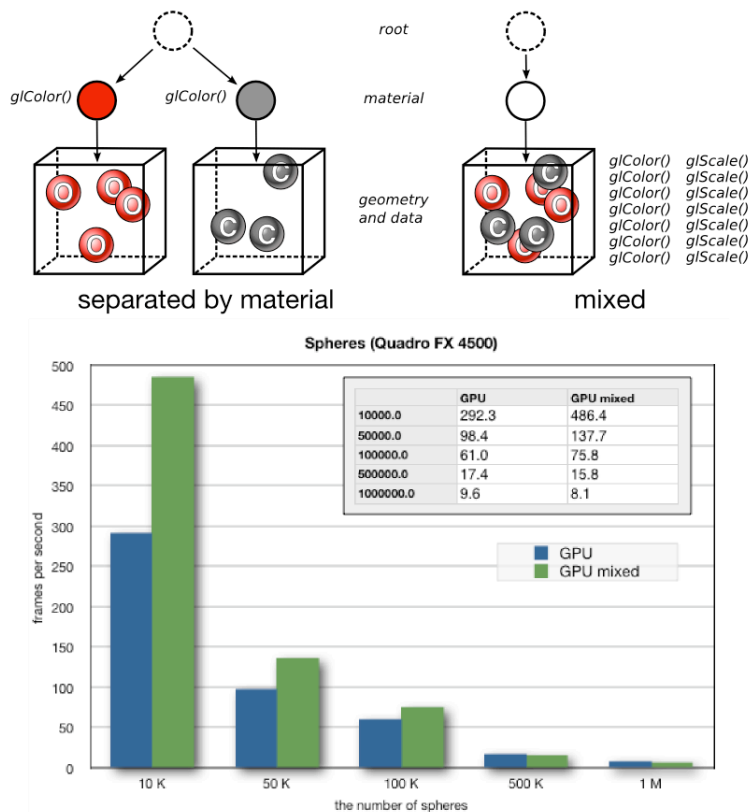


Figure 6.6: The material test is focused on the performance of OpenGL as a state machine. Lower numbers of state changes often lead to higher framerates. The image shows two scene graphs with different designs on the top. The first scene graph consists of multiple sphere nodes. Each sphere node is dedicated to a single species (Oxygen and Carbon) with specialized geometry and material properties. All atoms in one node are rendered inside a single loop with minimal changes in OpenGL states. The number of the loops depends on the number of the nodes. Introduction of multiple loops is the cause for less effective approach for small data sets. The second scene graph is limited only to one sphere node. The sphere node contains all atoms of all species. It is necessary to setup appropriate color and a scale for each atom separately. The performance of the scene graphs is compared in the graph. The number of the spheres vary from ten thousands up to one million. The spheres are divided between five materials. The first scene graph is more efficient for large data sets (half million of the spheres and more). The efficiency of the first scene graph is more notable with the other rendering methods, because they are using polygon meshes which need to be scaled for each sphere. Automatic normalization of the normals is furthermore required for hardware lighting. The optimized solution with multiple nodes provides better results in observable area of rendered frames (below 24 frames). More frames can not be seen.

## 6.2.4 Bonds

The test (*Fig. 6.7*) examines if texturing of the bonds can be used without a penalty.

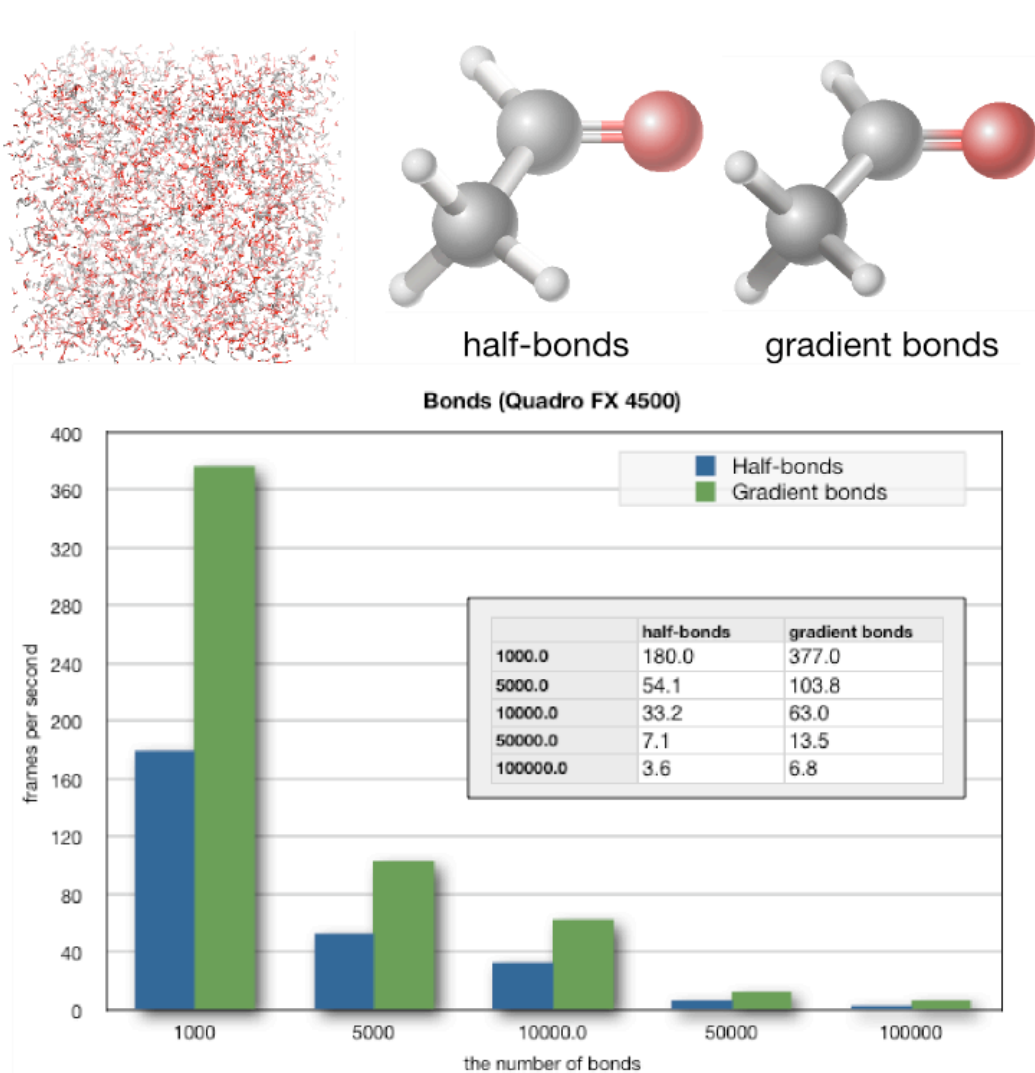


Figure 6.7: The graph shows the number of rendered frames per second for various numbers of the bonds which are visualized with two techniques. The bonds are generated inside a box from the atoms with random coordinates (top-left corner). The box contains hydrogens and oxygens, but the atoms are not visualized. The half-bonds are composed from two cylinders of different colors. The gradient bonds are represented with a single textured cylinder. As it can be expected, to render one cylinder is faster then to render two of them. Additional texture coordinates in polygon meshes of gradient bonds and usage of texturing units do not introduce serious slowdowns.

### 6.3 Postscript

Postscript renderer allows to take high quality screenshots in vector format. The downside of Postscript outputs (*Fig. 6.8*) is shown in the table (*Fig. 6.9*).

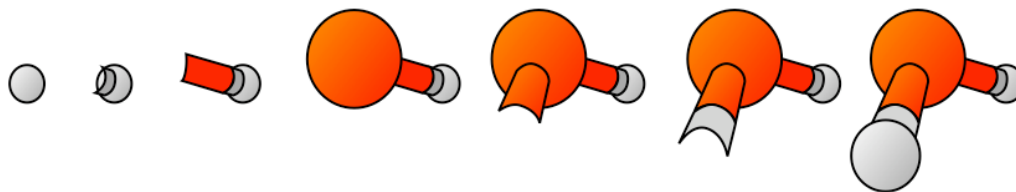
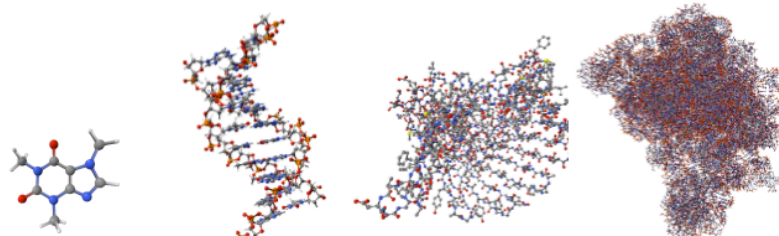


Figure 6.8: The image shows a molecule of water composed from Postscript curves. The curves are drawn from back to front (painters algorithm, left to right on the image). This is the principle of Postscript rendering. As it can be seen, the number of shapes will influent the size of Postscript files.



	caffeine	dna	porin	1N8R
spheres	24	637	2219	90418
cylinders	50	1370	4516	196508
*.ps	64 KB	1.6 MB	5.5 MB	234.4 MB
*.pdf	120 KB	3.2 MB	10.4 MB	
time		1 s	4 s	20 s

Figure 6.9: The table shows linear dependency of the file size according to the number of shapes. The limits are shown on a molecule of 1N8R where the output file starts to be too large. Rendering of large molecules can benefit from another vector formats like svg which support instantiating. The other way how to decrease the size is to simplify the gradients.

## 6.4 Hardware requirements

The test provides the insight to hardware requirements. Each configuration uses the best available rendering method. Three hardware configurations were tested:

- ATI Mobility Radeon 9500 (OpenGL 1.5, 32MB VRAM), 1GHz G4, 512MB RAM, Mac OS X 10.4.9
- GeForce4 Ti 4200 (OpenGL 1.5, 64MB VRAM), 2.4GHz P4, 512MB RAM, SUSE Linux 9.3
- Quadro FX 4500 (OpenGL 2.0, 512MB VRAM), 2x3GHz Dual-Core Xeon, 4GB RAM, Mac OS X 10.5.2

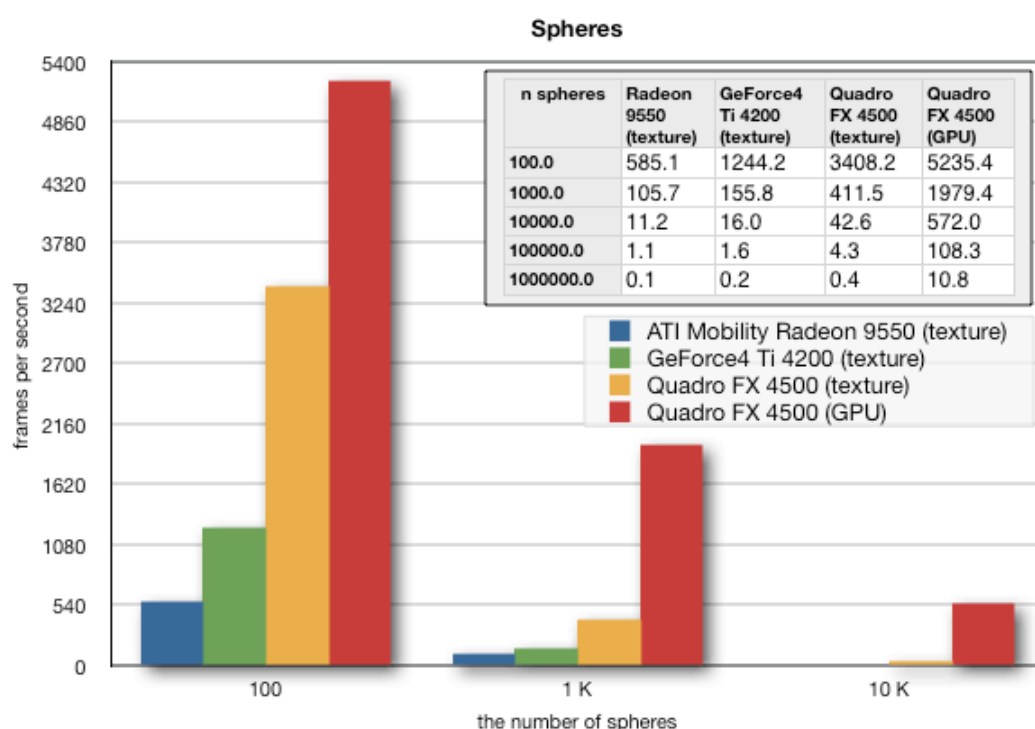


Figure 6.10: The graph shows the number of frames depending on the number of rendered spheres for three hardware configurations. The first three columns contain the values which were measured with the same rendering method (textured spheres). This was necessary, because the first two configurations do not support shaders. The results from shaders were added for reference (GPU column). Low-end graphics cards represented with ATI Mobility Radeon are capable of rendering up to ten thousands of the spheres at interactive speeds. Low-end graphics cards are supported and ATI Mobility Radeon is even a notebook class graphics card. Quadro FX with shaders is almost 100 times faster for one million atoms than the best rendering method available on ATI Mobility Radeon. This motivates the usage of modern graphics cards for visualization of atomic structures.

## Chapter 7

# Conclusion

The goal of this work was to develop an application for accurate and intuitive modeling of atomic structures using suitable modeling methods. The goal was fulfilled. No other program is available that comes anywhere near to proposed solution.

Available literature and software products focused on molecule visualization and modeling were studied. Editing of the atomic structures is usually quite complicated in the existing applications. Direct manipulations without the constraints are not precise. Accurate values can be provided only with the keyboard which is used to fill the dialogs. CrystalMaker is the state-of-the-art editor of atomic structures. It has a lot of features, direct manipulations, good visual quality and it is simple to use thanks to the standard Photoshop like user interface.

A suitable visualization was obtained using the scene graph approach. One geometry node can contain multiple coordinates in order to render multiple instances of the same object. The scene graph and the nodes were designed to support the interactions with the scene. This is accomplished with transformations in rendering pipeline instead of translating or rotating the coordinates in the main memory. Stereo rendering is available to fully exploit our visual ability.

New library of widgets for manipulation with the objects in the scene was designed. The library creates 3d graphical user interface specialized for modeling of the atomic structures. The components are created with a composition of basic interactive elements. The atoms, the bonds and helping points can be picked to constrain the widgets in order to provide accurate transformations. Advanced input devices with six degrees of freedom and gestures created from simultaneous multiple inputs can be used to manipulate the scene.

Demonstration software was implemented in a form of cross platform application. Supported platforms include Microsoft Windows, Linux and Mac OS X.

The most difficult task was to find the best representation of the atomic structures. The final design is created with two separated parts. The first part is used for visualization and the second for editing. The other demanding task was to make the gizmos simple to use together with their appropriate visualization in the scene.

The future work will be oriented towards better organization of the objects in the scene using layers and groups. The rendering performance can be improved with linear approach instead of visiting each node in the scene graph. The most effective rendering methods of the nodes were achieved with programable rendering pipeline using the shaders. The fragment programs can be more optimized in the future since they represent the current bottleneck in rendering performance. New widgets, input devices and gestures can be added to extend the user interface.





# Bibliography

- [1] Aylward S. a Barre S. a Cedilnik A. a Cole D. a Geveci B. a Hoffman B. a Ibanez L. a Schroeder W. a Squillacote A. a Yuan Y. Kitware's software developers' quarterly. [http://www.kitware.com/products/newsletter/kitware\\_quarterly0107.pdf](http://www.kitware.com/products/newsletter/kitware_quarterly0107.pdf), January 2007.
- [2] Accelrys. Materials studio - modeling and simulation solutions for the chemicals, materials, and pharmaceutical industries. <http://www.accelrys.com/products/mstudio/>, December 2007.
- [3] AMD. Firestream 9170 specifications. <http://ati.amd.com/products/streamprocessor/specs.html>, April 2008.
- [4] Autodesk. Autodesk - autodesk 3ds max. <http://www.autodesk.com/3dsmax>, December 2007.
- [5] Sigg Ch., Weyrich T., Botsch M., and Gross M. Gpu-based ray-casting of quadric surfaces. <http://www.cs.princeton.edu/~tweyrich/projects/quadrics/pbg06.pdf>, 2006.
- [6] Chimera. Ucsf chimera home page. <http://www.cgl.ucsf.edu/chimera/>, December 2007.
- [7] Max N. Crawfis R.A. Texture splats for 3d scalar and vector field visualization. *IEEE Visualization*, 1993.
- [8] CrystalMaker. Introduction and overview. <http://www.crystallmaker.com/crystallmaker/index.html>, December 2007.
- [9] Chris Gilbert. Nehe productions: Opengl article 6. <http://nehe.gamedev.net/data/articles/article.asp?article=06>, April 2006.
- [10] Elias H. Fast phong shading. [http://freespace.virgin.net/hugo.elias/graphics/x\\_polyp2.htm](http://freespace.virgin.net/hugo.elias/graphics/x_polyp2.htm), December 2007.
- [11] Reusch W. H. *An Introduction To Organic Chemistry*. Holden-Day Inc., 1977. ISBN 0-8162-7161-5.
- [12] Kilgard M. J. Avoiding 16 common opengl pitfalls. <http://www.opengl.org/resources/features/KilgardTechniques/oglpitfall/>, December 2007.

- [13] Schroeder W. J., Martin K., and Lorensen B. *The Visualization Toolkit An Object-Oriented Approach To 3D Graphics*. Kitware, Inc. publishers, 1998. ISBN 1-930934-19-X.
- [14] Wernecke J. *The Inventor toolmaker : extending Open Inventor, release 2*. Addison-Wesley, 2002. ISBN 0-201-62493-1.
- [15] Kitware. Vtk home page. <http://www.vtk.org/>, December 2007.
- [16] Westover L. Interactive volume rendering. *Proceedings of the Chapel Hill Workshop on Volume Visualization*, 1989.
- [17] Tarini M., Cignoni P., and Montani C. Ambient occlusion and edge cueing to enhance real time molecular visualization. [http://vcg.isti.cnr.it/Publications/2006/TCM06/Tarini\\_FinalVersionElec.pdf](http://vcg.isti.cnr.it/Publications/2006/TCM06/Tarini_FinalVersionElec.pdf), September 2006.
- [18] MolWorks. Integrated software tool for molecular design. <http://www.molworks.com/en/>, December 2007.
- [19] Mueller K. Neophytou N. Gpu accelerated image aligned splatting. *Volume Graphics*, 2005.
- [20] OpenGL. Glut - the opengl utility toolkit. <http://www.opengl.org/resources/libraries/glut/>, December 2007.
- [21] Bourke P. Calculating stereo pairs. <http://local.wasp.uwa.edu.au/~pbourke/projection/stereorender/>, December 2007.
- [22] Bourke P. Hershey vector font. <http://local.wasp.uwa.edu.au/~pbourke/dataformats/hershey/>, April 2008.
- [23] WWW pages. Online unit conversion. [http://www.onlineunitconversion.com/bicron\\_to\\_bohr.html](http://www.onlineunitconversion.com/bicron_to_bohr.html), April 2008.
- [24] PyMOL. Pymol users manual. <http://pymol.sourceforge.net/newman/user/toc.html>, December 2007.
- [25] Allouche A. R. Gabedit: A graphical user interface to computational chemistry packages. <http://gabedit.sourceforge.net/home.html>, December 2007.
- [26] Trolltech. Qt cross-platform application framework. <http://trolltech.com/products/qt>, December 2007.
- [27] Vasilyev V. Java molecular editor/builder home. <http://sf.anu.edu.au/~vvv900/cct/appl/jmoleditor/>, December 2007.
- [28] VMD. Vmd user's guide. <http://www.ks.uiuc.edu/Research/vmd/current/ug/ug.html>, December 2007.
- [29] Brodlie K. W., Carpenter L. A., Earnshaw R. A., Gallop J. R., Hubbold R. J., Mumford A. M., Osland C. D., and Quarendon P. *Scientific Visualization, Techniques and Applications*. Springer-Verlag Berlin Heidelberg, 1992. ISBN 0-387-54565-4.

- [30] W3C. Objects, images, and applets in html documents.  
<http://www.w3.org/TR/html4/struct/objects.html#h-13.6>, April 2008.
- [31] Rogers Y., Sharp H., Benyon D., Holland S., and Carey T. *Human-Computer Interaction*. Addison-Wesley, 1994. ISBN 0-201-62769-8.

# Appendices

## .1 Reference machine

The reference machine that was used in all experiments:

- **CPU:** 2 x 3 GHz Dual-Core Intel Xeon
- **RAM:** 4 GB 667 MHz DDR2
- **Graphic:** NVIDIA Quadro FX 4500, 512 MB VRAM, OpenGL 2.0
- **OS:** Mac OS 10.5.2
- **Compiler:** GCC version 4.0.1

## .2 CD

- Source code.
- User's guide.
- Program documentation.
- Video tutorials.

# Appendix A

## Screenshots

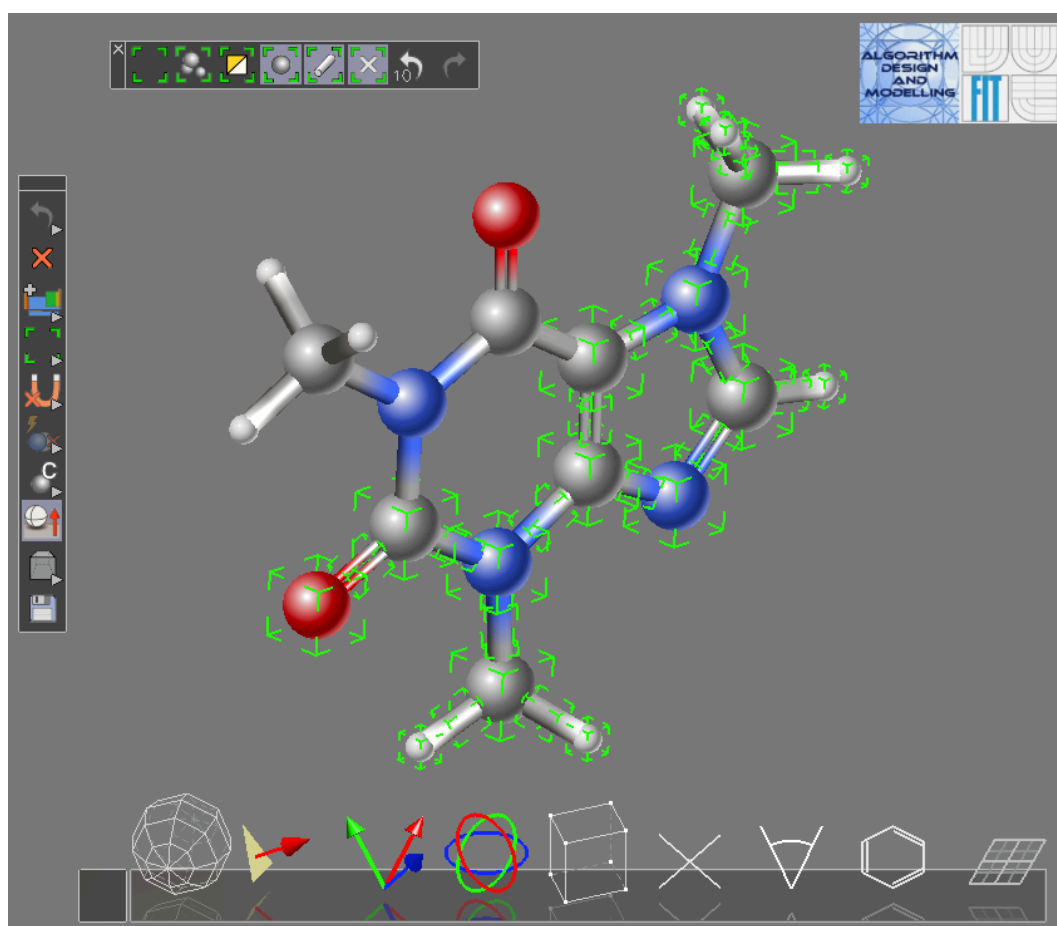


Figure A.1: Partially selected molecule of caffeine.

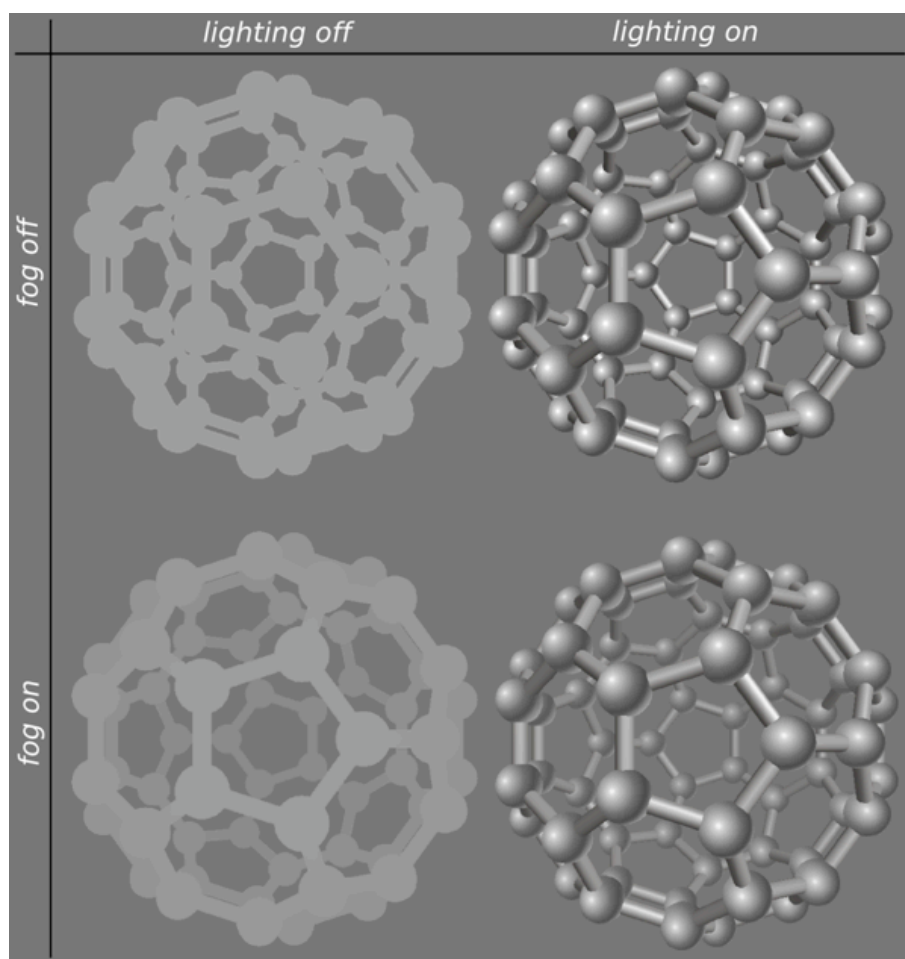


Figure A.2: The image shows four molecules of C<sub>60</sub> fullerene displayed with different lighting and fog parameters in perspective projection. 3d nature of the structure is best perceived when both lighting and fogging are enabled.

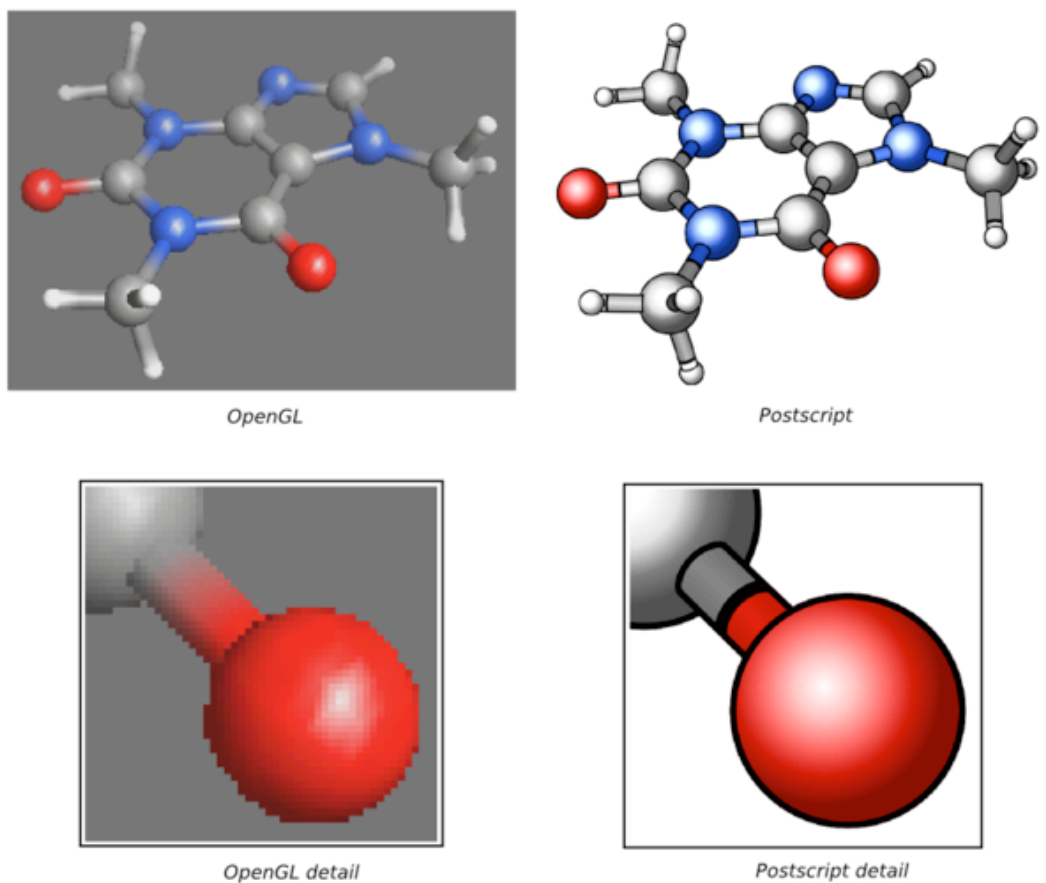


Figure A.3: The quality of OpenGL (raster) and Postscript (vector) outputs is compared on the image.



# Appendix B

## Class Hierarchy

- ArAABBox
- ArApp
- ArAtomDef
- ArAtomicStructure
- ArBMP
- ArBonds::ArBondInfo
- ArBondsNodes
- ArBondsNodes::ArBondCombo
- ArCamera
- ArCollisions
- ArCommands
- ArCommands::ArCommand
  - ArCommands::ArChangeSpecies
  - ArCommands::ArCreatePoint
  - ArCommands::ArRemove
  - ArCommands::ArTransform
  - ArCommands::ArTranslate
- ArConsumerGL
  - ArFontStrokeGL
  - ArMeshGroupGL
  - ArNodeBond
  - ArNodeSphere
  - ArShader

- ArSplats
  - ArTextureGL
  - ArVertexBuffer
- ArDevice
  - ArDeviceMouse
  - ArDevicePatriot
- ArElements
- ArEvent
- ArFastPhong
- ArGauss
- ArGeoSource
  - ArGeoSourceBox
  - ArGeoSourceCone
  - ArGeoSourceCylinder
  - ArGeoSourceDisk
  - ArGeoSourceSphere
- ArGizmoImageMap::ArGizmoImageShape
- ArGizmoPoints
- ArGL
- ArGridMap3  $\langle T \rangle$
- ArGroup
- ArHit
- ArImage
- ArImageColors
- ArImageHistogram
- ArIntersection
- ArKey
- ArLOD
- ArManagerGL
- ArManagerGL::ArResource
- ArMath  $\langle T \rangle$

- ArMatrix3n  $\langle T \rangle$
- ArMatrix4  $\langle T \rangle$
- ArMeanShift
- ArMesh
- ArMeshData
- ArMeshXML
- ArNames
- ArNode
  - ArGizmoMeasureGroup
  - ArGizmoPlaneGroup
  - ArMaterial
  - ArNodeCircles
  - ArNodeLine
  - ArNodeText
  - ArPrimitive
    - \* ArGizmo2d
    - \* ArGizmoArray
    - \* ArGizmoArrow
    - \* ArGizmoAtom
    - \* ArGizmoBox
    - \* ArGizmoCircle
    - \* ArGizmoDock
    - \* ArGizmoHand
    - \* ArGizmoIcon
    - \* ArGizmoImage
      - ArGizmoImageMap
    - \* ArGizmoInfPlane
    - \* ArGizmoMatrix3
    - \* ArGizmoMenu3d
    - \* ArGizmoPatriot
    - \* ArGizmoPicker
      - ArGizmoCPlane
      - ArGizmoMeasure
      - ArGizmoPlane
      - ArGizmoRotate
      - ArGizmoSnapPoint
      - ArGizmoTemplate
      - ArGizmoTranslate

- \* ArGizmoSphere
  - \* ArGizmoText
  - \* ArGizmoToolbar
- ArSelectable
  - \* ArNodeBond
  - \* ArNodePoint
    - ArNodeSphere
- ArSplats
- ArTransform
- ArNode::ArNodeObserver
  - ArAtoms
  - ArBonds
  - ArModule
    - \* ArEditor
      - ArModuleAtoms
    - \* ArModuleEvalMesh
    - \* ArModuleGenMesh
    - \* ArModuleLoadMesh
    - \* ArModuleLoadStructure
    - \* ArModuleSplats
  - ArModuleArPoints
- ArOctreeMap3 < T >
- ArPtr < T >
- ArQuaternion < T >
- ArSelections
- ArSelections::ArSelect
- ArSnap
- ArString
- ArThread
  - ArBonds::ArBondsTask
  - ArNodeBond::ArBondTask
  - ArNodePoint::ArOctreeTask
- ArTimeMachine
- ArTimer
- ArTimer::ArTimerObserver

- ArBonds
  - ArGizmoSnap
  - ArNodeBond
  - ArPrimitive
- ArUndoManager
- ArUVMapper
  - ArUVMapperConst
  - ArUVMapperFastPhong
- ArVector  $\langle T \rangle$
- ArVector3  $\langle T \rangle$
- ArVertexBuffer::ArVertexC4UBV3F
- ArVertexBuffer::ArVertexT2FC4UBV3F
- ArViewport
  - ArRenderer
    - \* ArRendererGL
    - \* ArRendererPS
- ArWindow
- ArXMLNode

## Appendix C

# Atom Editor: User's guide



## Visualization and Modelling of Molecules and Crystals


diploma project

AUTHOR Václav Bubník

SUPERVISOR Pavel Zemčík, doc. Dr. Ing.

CONSULTANT Sixten Boeck



Max-Planck-Institut  
für Eisenforschung GmbH   
Computational Materials Design

## C.1 Introduction

This document describes AtomEditor application. AtomEditor could be considered as a CAD application for the atomic structures. It allows very accurate editing of molecules, crystals or nanostructures.

With this tool you will be able to load and modify existing structures or to create new structures from scratch. The results can be saved to a file.

## C.2 Application

The application uses OpenGL and GLUT framework for rendering and communication with the operating system. This combination leads to good portability between different platforms and operating systems. Both unix systems (tested on Mac OS X and Linux) and Windows (tested on Windows XP) are supported.

The application can be also build with multithreading support **Pthreads** to take the benefits from multi-core and multi-CPU systems. Several rendering optimizations and bond algorithms can be distributed between multiple cores.

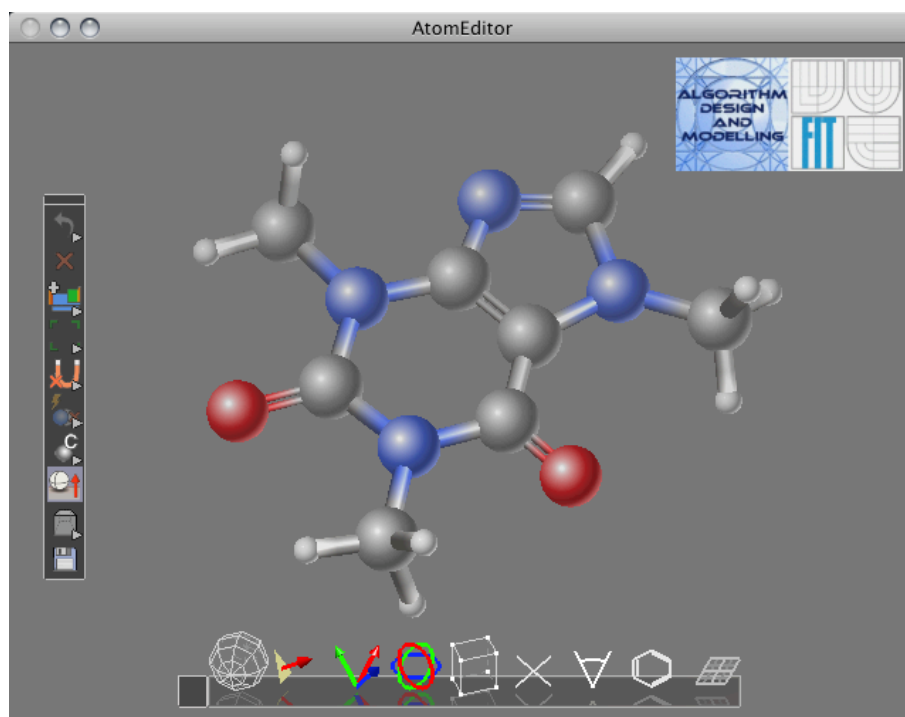


Figure C.1: A screenshot of application window can be seen on the image. A molecule of caffeine is visualized with the application.

The application is a hybrid between command line and window application. Even that the application has a graphical user interface [C.4](#) the input filename is provided from the command line during program execution together with other optional parameters.

To compile the application please read the file README. After the compilation you may run atom editor with following options:

- `./atoms` ..... to build a new structure from scratch
- `./atoms -h` ..... prints short help with examples
- `./atoms -i filename ...` to load an existing structure, **PDB** and **XYZ** file formats are supported
- `./atoms -s` ..... to enable Quad-buffer stereo rendering (tested with **CrystalEyes** shutter glasses). The stereo mode is the most useful with perspective viewports.

## C.3 Controls

Possible input devices and methods of their interactions are presented in this chapter.

### C.3.1 Mouse

The main control device is a mouse. The mouse is used for menu selections, object selections, manipulation with the camera and interaction with the gizmos. The gizmos are interactive elements in the scene with specialized functionality.

There are two mouse modes: passive and active. All mouse buttons are released in the passive mode. Moving the mouse cursor in the passive mode shows various feedbacks from the system: the icons in the toolbars get highlighted, 3d icons in the docks and the icons in the scene are magnified and the objects (atoms, bonds, and points) in the scene are outlined with highlighted colors as the mouse cursor enters their areas. We can say that whenever some kind of interaction is possible, it is visualized and we can see it. You may already know this style of passive interaction from adventure games like Monkey Island.

One of the mouse buttons is pressed in the active mouse mode. This is the active interaction where you move the atoms, click the icons, select a new gizmo etc. The interaction ends when the button is released.

The left mouse button is used to select an object under the cursor or to rotate (or pan with shift) the camera when there are no objects under the cursor. The right mouse button is used to zoom the camera or to scale a gizmo. Only the sphere gizmo and the docks can be scaled.

### C.3.2 Keyboard

The keyboard is used to change the number of viewports and few gizmos respond to keyboard inputs. The camera can be moved along the view plane (pan) when the shift key modifier is hold while moving the mouse with the left button pressed.

The editor is ready to receive the commands from a command line like *select all*. This feature will be implemented in the feature version.



## Keyboard shortcuts

- Alt + '1'—'2'—'3'—'4' ... set the number of viewports
- Alt + 'r' ..... Postscript preview
- Shift + mouse ..... camera pan

### C.3.3 Motion tracking devices

Only one motion tracking device is supported at this time as a prove of concept. We have chosen **Patriot** device with 6DOF. The drivers for the device are available for Windows, Linux and we developed our drivers for Mac OS X.

Multi-touch and other HW is easily to support as a derivate of this solution.

The application supports usage of two Patriot sensors at the same time. The first sensor (right hand) is used for interactions with the scene. The second sensor (left hand) is used for manipulations with the camera.

#### Single sensor modes

- the first sensor .... interaction with the scene and gui
- the second sensor ... manipulation with the camera

The right hand sensor can be used in the left hand mode and vice-versa. This approach extends camera and scene manipulations to two sensors. The sensors are visualized in the scene with the different colors in a single mode. The second sensor enters combined mode as long as the button on the first sensor is hold. The sensors are visualized with the same colors in combined mode.

#### Combined modes

- camera ... the second sensor controls rotation of the camera
- scene .... the distance between the sensors determines scale

#### Gestures

- lock to zoom ..... by aligning the camera sensor with the z axis
- camera rotation ... the sense of rotation is reversed in negative hemisphere determined by the z axis
- scale ..... the distance between the sensors



See video tutorial 11. Motion tracking input device for practical demonstration.

## C.4 GUI

Graphic User Interface consists of the toolbars, the docks, the tooltips and the gizmos. The dock is a container with gizmos and will be described more in details later. The interface is rendered in 3d space for better visibility in stereo mode.

### C.4.1 Toolbar

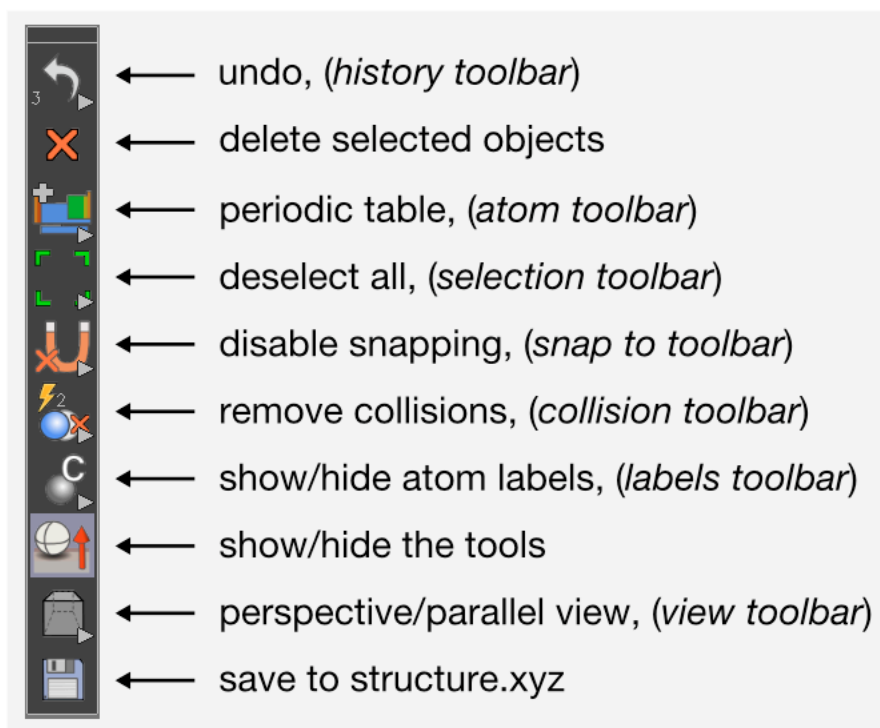


Figure C.2: The image shows the main toolbar with all tooltips.

The main toolbar is a substitute for menu. It is faster to navigate than a menu and frequently used sub-toolbars (sub-menus) can stay opened in the workspace. All toolbars are floating.

Click the left mouse button to trigger a command of the icon under the cursor.

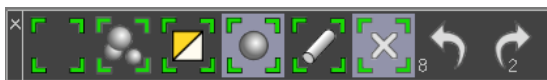
Some icons can be disabled (shaded) if they can not be applied. For example the icon *delete selected objects* will be disabled if there are no objects selected in the scene. You can still access the sub-toolbars from disabled icons.

This type of user interface is well standardized across various CAD and modeling systems.

Some items in the menu have sub-toolbars with more commands related to the same group. The presence of a sub-toolbar is visualized with the arrow in the bottom-right corner of the icon. Simply hold the left mouse button for about 0.5 second to access the sub-toolbar of the item under the cursor. The sub-toolbar automatically disappears after it is used if you did not detach it.






## Selection toolbar






The commands dealing with selections are grouped in this toolbar.



The toolbar can be divided to three parts. The first part contains selection commands:

-  Deselect all objects.
-  Select all objects.
-  Invert the current selection.

The second part contains selection filters. A filter is in allow state when it is highlighted. Take the picture of selection toolbar as the example where selection of the bonds is denied and all other selections are allowed.

-  Allow/deny selection of the atoms.
-  Allow/deny selection of the bonds.
-  Allow/deny selection of the points.

The third part controls the history of selections. The history is limited to the last ten selections. This can be quite useful since some commands deselect all objects used as their input.

-  Previous selection.
-  Next selection.






See video tutorial 2. Selection for practical demonstration.

## Snap to toolbar



Snapping is applied in free move mode where you can snap to the points or to the atoms. Snapping to the atoms results in collisions [C.4.1](#) since there will be two (or more) atoms with the same coordinates.


-  Disable snapping.
-  2d snapping projected to the view plane.
-  3d snapping.





See video tutorial 4. Snap or 5. Collisions for practical demonstrations.

## Collision toolbar



The atoms in collision are emphasized with red clouds and  bolts. A collision occurs if there are two (or more) atoms with the same or very close coordinates. The total number of atoms in collision is shown in the icons.

There are two ways how to solve all collisions:

-  Keep only the last modified atom in each collision.
-  Keep only the oldest atom in each collision.





See video tutorial 5. Collisions for practical demonstration.

## Labels toolbar



The labels display chemical symbols of the atoms.

-  Show/hide atom labels.
-  Show/hide atom numbers as the were read from the input file or created. These numbers express the order of the atoms.





See video tutorial 1. Camera for practical demonstration.

## View toolbar



The target of these commands is always the current viewport.

-  Switch between perspective and parallel views.
-  Export the current viewport to Postscript file *screen.ps*.



See video tutorial 6. Export to Postscript for practical demonstration.

## C.4.2 Dock

The dock is a sort of toolbar except that the icons are 3d objects. Each icon represents a gizmo that can be enabled or disabled. Enabled gizmos have yellow icons. The gizmos are interactive elements used to manipulate the scene. 3d icons are preferred instead of 2d icons to achieve better mapping with graphic representations of the gizmos.

Each gizmo require some user interaction in the scene. This is different from commands triggered from the toolbars.

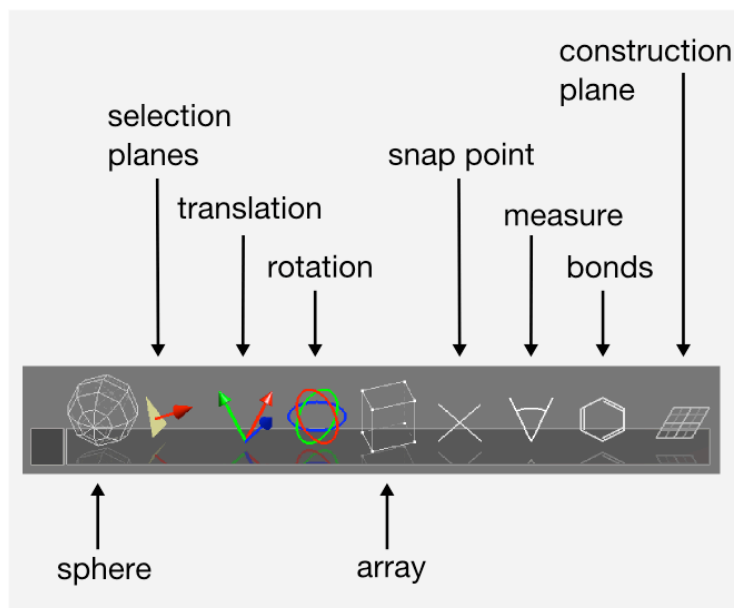


Figure C.3: The tools available in the dock. The gizmos that can be further modified with picking have their descriptions on the top.

### Sphere gizmo

Sphere gizmo is designed for volumetric selections. Only the objects (atoms/bonds/points) inside the sphere are selected. The sphere can be moved (left button) and scaled (right button) using the mouse. The origin of the sphere is snap-able to the atoms and to the points for easier manipulation in 3d space.

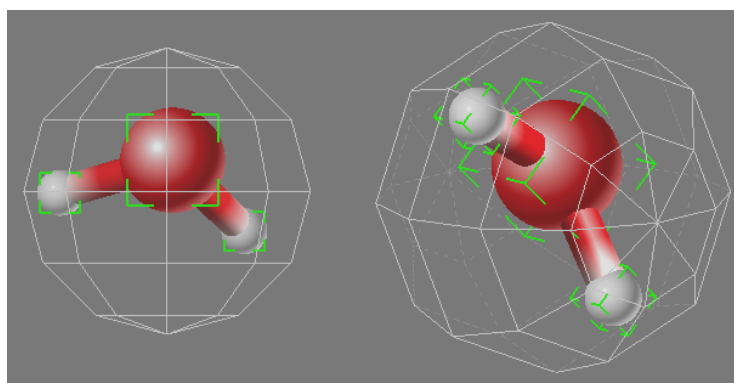


Figure C.4: A molecule of water is selected with the sphere gizmo.

This gizmo can be used with other gizmos simultaneously. The radius can be specified from the keyboard when the gizmo is focused with the mouse.



See video tutorial 2. Selection or 10. Translate for practical demonstrations.

## Plane gizmo

The planes are designed for volumetric selections. You may create multiple planes to select all objects (atoms/bonds/points) between them.

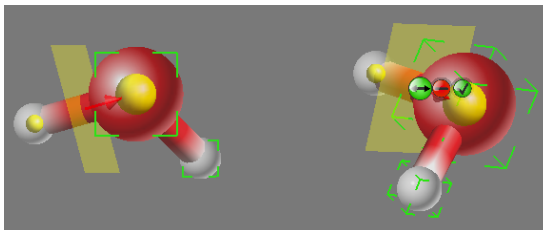





Figure C.5: Plane gizmo is used to select two atoms.

The plane can be set picking the atoms, the bonds or the points. You can place the plane between two objects or to the plane created from three picked objects. You may also pick a bond to set the plane between the atoms on the ends.

Each plane has a menu. You can create new planes  or delete the existing ones  from the menu. Plane normal can be flipped  to select the opposite half-space.



See video tutorial 2. Selection for practical demonstration.

## Translation gizmo

The main purpose of this gizmo is to create constrained movements. The secondary function is to copy selected atoms along predefined translation.

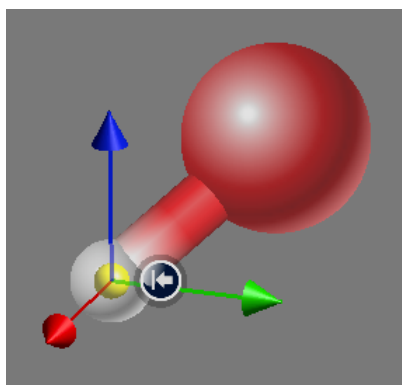


Figure C.6: Translation gizmo.

The movements are constrained in two ways. The first is determined by the coordinates of picked objects (atoms/bonds/points). The second constrain is snapping along that direction with multipliers of  $1/4$  of it's length.



Up to three coordinates taken from the atoms/bonds/points can be picked to define the first constrain. A bond can be picked to use the coordinates of atoms on the ends.

- There is no constrain from a single picked coordinate.
- 2 picked coordinates mean translation along the line between two points.
- 3 picked coordinates mean translation in the plane defined by three points or translation along the normal of that plane.



The copy function shows a preview of the atoms that will be created soon as the mouse enters the copy icon.



It is possible to reset all picked objects using the reset icon.



See video tutorial 10. Translate for practical demonstration.

### Rotation gizmo

This gizmo is designed to rotate selected atoms.

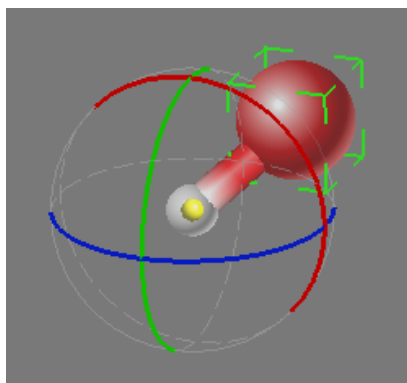


Figure C.7: Rotation gizmo.

The rotations are constrained in two ways. The first is determined by the coordinates of picked objects (atoms/bonds/points). The second constrain is snapping along the angle with multipliers of 30 degrees. Snapping constrains can be set precisely from the angles measured with Measure gizmo.

Up to three coordinates taken from the atoms/bonds/points can be picked to define the first constrain. A bond can be picked to use the coordinates of atoms on the ends.

- 1 picked coordinate means rotation around that point
- 2 picked coordinates mean rotation around the line between two points
- 3 picked coordinates mean rotation in the plane defined by three points



See video tutorial 7. Measure, rotate for practical demonstration.

## Array gizmo

The array is simplified crystal builder. The snap points in the corners of each cell are created automatically. The array can be used with other gizmos simultaneously.

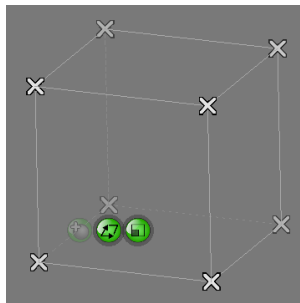

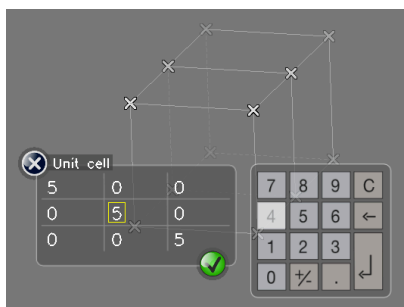


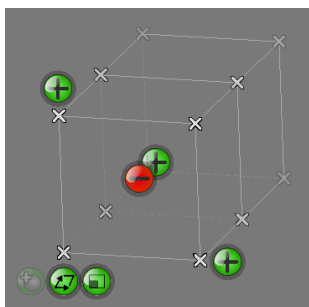
Figure C.8: Array gizmo with the single cell.


There are four steps to build a new crystal:

-  Setup the unit cell matrix. You may also use the keyboard to enter the values.



-  Set the number of cell repetitions along the axes using plus  and minus .



- Select some atoms that will be repeated. These atoms are usually inside the first unit cell.
-  Click on copy icon. This icon also features with a preview before the command is actually triggered. The icon is enabled only if there are some atoms selected.



See video tutorial 8. Array or 12. Points, Array for practical demonstrations.

### Snap point gizmo

Snap point gizmo creates new snap points.

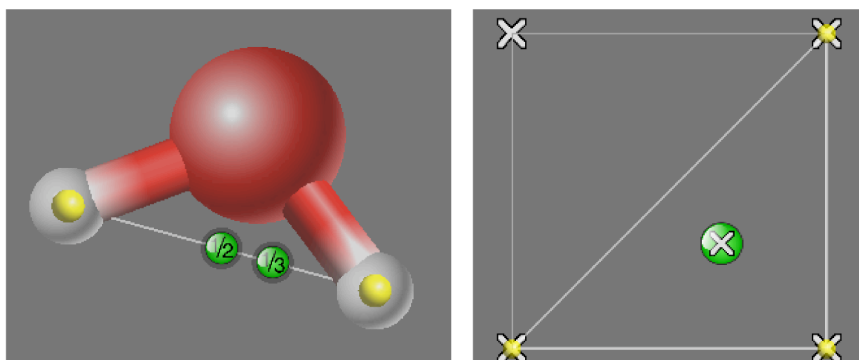


Figure C.9: Snap point gizmo is defined between two hydrogens (left). Three points from the array gizmo were picked to define the gizmo (right)

Up to three coordinates taken from the atoms/bonds/points can be picked to define the gizmo. A bond can be picked to use the coordinates of atoms on the ends.

- 2 picked coordinates allows to create new point in one half or one third of the distance between the coordinates.
- 3 picked coordinates allows to create new point in the center of the coordinates.



See video tutorial 12. Points, Array for practical demonstration.

### Measure gizmo

Measure gizmo can be used to measure distances, angles and dihedral angles between picked objects. It is possible to create multiple measurements. The measured values are dynamic. The values are automatically updated as the picked objects change their coordinates.



Figure C.10: A molecule of water with measured angle can be seen on the left. The right part of the image shows the local menu of the gizmo. The menu can be accessed by clicking on the measured value and closed with the x symbol. The menu allows remove the measured value (minus) or to create additional measurements (check).

Up to four coordinates taken from the atoms/bonds/points can be picked to define the gizmo. A bond can be picked to use the coordinates of atoms on the ends.

- Pick two coordinates to measure the distance between them.
- Pick three coordinates to measure the angle between them.
- Pick four coordinates to measure dihedral angle between them.

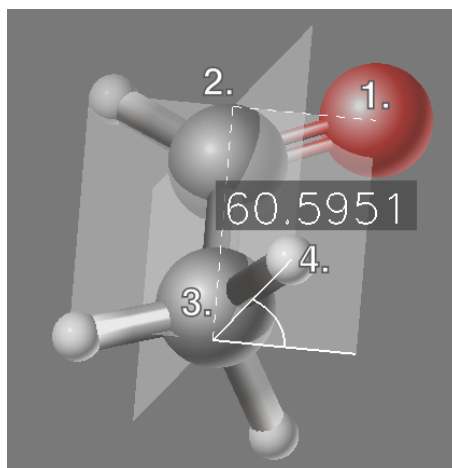





Figure C.11: The image shows a molecule of acetaldehyde with measured dihedral angle.

Each measurement has a menu. Click on the measured value to access the menu. You can create new measurements  or delete the existing ones  from the menu. The menu can be closed with  close icon.



See video tutorial 7. Measure, rotate for practical demonstration.

### Bonds gizmo

Bonds gizmo can create new bonds and remove or change existing bonds.

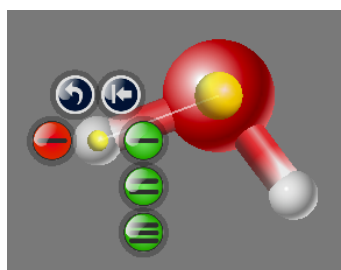


Figure C.12: Bonds gizmo.

To use the gizmo:

- First pick at least two atoms. You may pick more of them to create a path.



- Use minus to delete all bonds along the path between picked coordinates.



- Click on one of the icons with specified bond order (single, double, triple) to create new bonds or to modify existing bonds along the path.



- Remove the last picked coordinate from the path.



- Click on this icon in order to reset the gizmo.



See video tutorial 9. Bonds for practical demonstration.

### Construction plane gizmo

The main functionality of this gizmo is to setup the camera precisely. The secondary functionality is to define a construction plane.

The construction plane can be created only from three picked coordinates. Created grid provides snapping constraints for the atoms. The construction plane can be used with other gizmos simultaneously since it remains visible even when the gizmo is disabled.

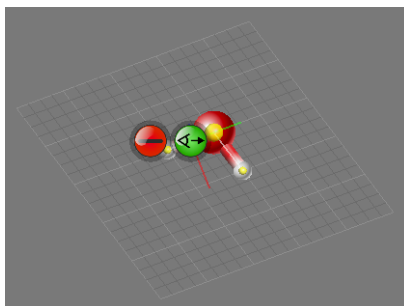


Figure C.13: Construction plane defined in H<sub>2</sub>O molecule.



- This icon removes existing plane. The icon is hidden if there is no plane.



- Click on this icon to setup the camera from picked coordinates.



- Click on this icon to create new construction plane. The icon features with a preview of that plane. It is necessary to pick three coordinates to access this icon.



See video tutorial 10. Translate or 1. Camera for practical demonstrations.

### C.4.3 Elements

This gizmo allows quick modifications of the species or creation of new atoms from selected snap points.

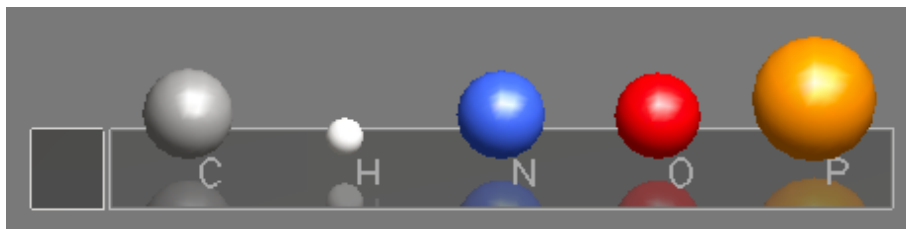


Figure C.14: The elements used in a DNA: Carbon, Hydrogen, Nitrogen, Oxygen and Phosphor. 3d representations of the atoms show relative atomic radii of the atoms in the palette.

Click the left mouse button on an atom from the palette in order to use the same species.

You may add new species using  the periodic table of elements [C.4.1](#).



See video tutorial 3. Change, remove, undo, redo for practical demonstration.

## C.5 Video tutorials

The video tutorials are mp4 files containing video streams encoded with H.264 codec at 10 frames per second and no audio. They should be playable directly from web browsers.

- Camera
- Selection
- Change, remove, undo, redo
- Snapping
- Collisions
- Export to Postscript
- Measure, rotate
- Array
- Bonds
- Translate
- Motion tracking input device
- Points, Array