Metaphors for software visualization systems based on virtual reality

Vladimir Averbukh[1,2][0000−0002−4379−1450], Natalya Averbukh[2][0000−0002−8232−6711], Pavel Vasev[1][0000−0003−3854−0670], and Ilya Gvozdarev[2][0000−0002−1896−1753], Georgy Levchuk[2][0000−0003−1484−778], Leonid Melkozerov[2][0000−0002−8921−1204], Igor Mikhaylov[2][0000−0003−4823−5944]

[1] Institute of Mathematics and Mechanics of the Ural Branch of the Russian Academy of Sciences, Ekaterinburg, Russia averbukh@imm.uran.ru
[2] Ural Federal University. Ekaterinburg. Russia

Abstract. The paper discusses research and development in the field of software visualization based on virtual reality environments. Spatial metaphors, in particular, a city metaphor and different versions of cosmic space metaphors, play an important role in such systems. The paper reviews such aspects of the theory of computer visualization and the theory of visualization metaphor as metaphor features.
A brief overview of the projects of software visualization systems based on virtual reality is provided. Among the systems developed over the past decades, one can find systems both for program visualization and for visual programming. Descriptions of prototypes of software visualization systems, software objects visualization and supercomputer performance data visualization, realized by the authors of the paper, are presented. These prototypes, designed for virtual reality environments, were developed with the use of several versions of a cosmic space metaphor and an extended city metaphor. The paper also discusses such psychological aspects of the human factor in developing software visualization systems with the use of virtual reality as presence, performance and several issues connected with control and navigation.
The goal of the research and development, discussed in the paper, is the search for approaches that will enable efficient use of virtual reality in solving complex problems, which software professionals come across.

Keywords: Software Visualization · Visual Programming · Virtual Reality · Visualization Metaphors, Human Factor · Presence

1 Introduction

Software visualization can be defined as an assembly of CGI and human-computer interaction techniques employed for better explanation of notions and efficient software maintenance, as well as for specification and software objects presentation in the process of program design and development. First visual programming and program visualization systems that can be categorized as software

visualization appeared in late 70s early 80s. The emergence of this field is connected with the practice and ideas of graphic representation of programs, based on extensive use of flowcharts (control flow graphs) and data flow graphs by programers. At that time, mass computerization looked imminent and graphic representation of programs seemed to ease software development and support significantly. Software visualization systems typically include the systems of visual programming and programming visualization systems. A visualization idea was put forward stating that program design, checkout and support should be carried out within a unified system with the same graphic representation of entities.

Research and development in the field of software visualization systems with the use of virtual reality have been conducted since early 1990s. Recent years have seen a new surge of interest in the development of such systems.

The paper discusses the implementation of spatial metaphors in software visualization systems using virtual reality. The success of visualization, to a certain extent, depends on the choice of metaphors, although the factors of implementation quality, the human factor and usability also play a significant role.

Provided below is the analysis of spatial metaphors used in software visualization systems based on virtual reality, the overview or existing solutions for these systems, and several issues connected with the human factor in this context.

2 Software visualization metaphors analysis

Let us define a visualization metaphor as a mapping from concepts and objects of the simulated application domain to a system of similarities and analogies, and generating a set of views and a set of techniques for interaction with visual objects. Computer metaphor is considered the basic idea of assimilation between interactive visual objects and model objects of the application domain. Its role is to promote the best understanding of the semantics of interaction and visualization, and also to determine the visual representation of dialog objects and a set of user manipulations with them. Visualization metaphors form the basis for the views of specialized visualization systems. Designing such views is a crucial part of a thorough design of the human factor related aspects of these systems [2].

Objects of the new metaphorical domain, the relationship between them and the possible actions in this domain have a number of properties, which we call metaphor properties. The success or failure of visualization systems depends on many factors. One approach to the evaluation of visualization involves examining the properties of visualization metaphors. We analyze the properties to consider the possibility of using metaphors for specific applications of software visualization. It is important to understand what objects may be represented with one or another metaphor. We need to analyze the opportunities of visualization metaphors (more precisely, the views based on the visualization metaphors) to represent large and huge volumes of data and details required to understand the program's operations [1].

Spatial metaphors can be used in visualization systems based on virtual reality. Further, we will consider the properties of several spatial metaphors, such as a city metaphor, a molecule metaphor, and cosmic metaphors. Note that a city metaphor and cosmic metaphors are used in a number of systems, listed both in the review of existing solutions and in the description of prototypes designed by us.

A city metaphor has such properties as **unlimited context, organization of inner structure, naturalness, and resistance to scaling**. Unlimited context means that when visualizing a large volume of data one can see the whole image and easily identify key locations. The use of a city metaphor presupposes structuring the input data by means of an internal city structure, with separate blocks, streets and buildings. The naturalness of metaphors leads to simplicity of spatial orientation and easiness of navigation. The changes in the scale of data representation within the city metaphor can be effected, in particular, by changing the altitude on which the observation point is located. In software visualization systems, within a city metaphor, transport highways are often used to represent control flows and data, as well as various connections between objects and parts of the program. In order to identify a programs features, in particular, its bugs, one can highlight them with color, use a separate shape or size.

When these metaphors are used in the systems based on **virtual reality** one can implement a flight over the city and enter separate rooms, where the necessary visual information is represented. An expansion of a city metaphor is suggested through adding active agents by inputting parameters into certain functions and methods. The agents can move about the city, determining the locations where they are used, changed, and the way the process of algorithm work plays out. This way, an extended metaphor creates such additional properties as the opportunity to observe software objects inside buildings or rooms, reflecting particular entities in the course of active agents moving through the city.

Now, let us consider the properties of a molecule metaphor that may also be used to visualize call graphs of parallel programs (see Fig. 1). This metaphor can be used in software visualization, for example, to visualize dynamic object relationships in Java programs.

Now we shall examine the properties of a molecule metaphor. This metaphor may support visualization of large volumes of structured data (unlimited context). Interpretation of a physical molecule metaphor and its modifications usually is simple and natural, although it requires the user to have (basic) knowledge of physics. The relationship between objects of a visualized model is also represented naturally by links between atoms (naturalness). The views based on a molecule metaphor are resistant to scaling (resistance to scaling). A metaphor supports selection of key elements, for example, through coloring or changing the size of the molecule elements and thickness of communications between them (key elements). Moving and navigating in graphical displays related to a molecule metaphor may be performed by flying around the molecule. There is the experience of visually entering a separate atom and seeing internal visual

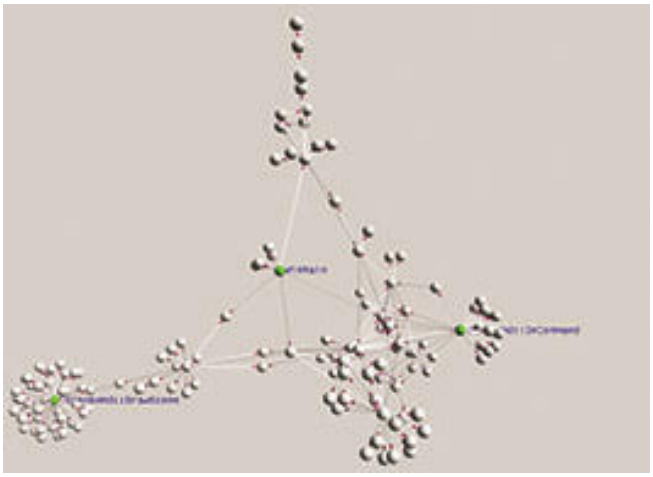

Fig. 1. Call graph visualization based on a molecule metaphor [4].

information inside a single sphere (see [3]). It is possible to implement similar entering spheres in the framework of virtual reality environments (organization of inner structure).

In a number of software visualization systems a cosmic metaphor with a heliocentric world view is used. Furthermore, entities may be represented as cosmic objects (stars and planets), as well as their satellites and other outer space elements.

Properties of this option of a cosmic metaphor may include:

- a context limited only by the possibilities of users spatial perception;
- freedom of choice in terms of entities location, with the opportunity of two-dimensional and three-dimensional representation in different situations;
- naturalness of the metaphor, which reduces the efforts required to interpret the resulting image;
- organization of inner structure: the structure of software (packages, classes and their fields and methods, relations between them, arguments etc.) can be projected onto the universe structure (a galaxy, a stellar system, a planet, satellites and rings);
- scalability.

This metaphor can also be used in designing software visualization systems based on virtual reality. Virtual reality will provide the advantages in navigation, movement through the structure of classes (outer space) and interaction. The stereoscopy of virtual reality means enables us to assess the relative size of objects and the distance between them, which will help eliminate the ambiguity of representation of complex entities [12].

Visualization based on this option of a cosmic metaphor can meet the quality criteria of visualization based on the analysis of visual data, including a sequence of repeated procedures of reviewing, scaling, filtering and detailing [27], supplemented by the procedures of educing relationships, acquiring search history and deriving subsets from the examined data [13].

In order to represent hierarchically distributed data one can use another cosmic metaphor option, relying on the ancient geocentric universe model. This model postulated the Earth being a static center of the universe. Around the Earth, transparent hard spheres revolve with celestial bodies attached to them: first, the Moon, then the Sun, then planets of the Solar system (each planet in its own sphere), and then stars. The whole model comprised a set of embedded spheres.

This metaphor enables us to present large volumes of multilevel data. In the case of virtual reality application, this metaphor provides good navigation and movement within the virtual world. A two-dimensional and a three-dimensional representation of the geocentric model existed, and for the purposes of software objects visualization we can use both options.

3 Software visualization based on virtual reality environments. Review of existing solutions

In early 90s an interesting software visualization system, Avatar, was developed [21], which makes extensive use of the means of virtual reality and a 3D version of a room building metaphor, and functions on the basis of a CAVE-type virtual reality environment. The Avatar system was designed to represent large volumes of data regarding the performance of parallel systems, acquired directly in the course of their work, which was used for performance checkout of parallel programs. In the process of work a user sort of finds themselves inside a 3D room, where a video image is projected on the walls. (In this context, the term *scattertube* is used in the system, which is derived from a traditional *scatterplot* image view.) The inner planes show axes, and the cube faces the floor and the walls of a room show the graphs describing performance metrics of parallel programs. Performance data on parallel programs were displayed as traditional two-dimensional graphs. A transparent mode for the ceiling and walls was implemented. The unification of separate elements (scattertube matrix) was realized, which resembles a glass skyscraper, with each of the rooms containing a 3D output, describing various aspects of behavior of a parallel program. A history timeline visual display is introduced as an alternative to a phase plane. A trip (virtual flight) about the skyscraper based on this timeline made it possible to research the data regarding the performance of an applied parallel program. Note that, despite a very interesting implementation of the Avatar system, the data regarding its use or development continuation were not to be found. This is, probably, either due to insufficient information capacity of the data used in visualization regarding two-dimensional graphs processors performance, or due to possible unpleasant and even painful sensational manifestations, which

a user could experience during a virtual flight along the *scattertube* that goes in accordance with the history timeline of a multiprocessor system work. These manifestations of **cybersickness** (a health disorder similar to seasickness) are often connected with the lack of possibility to actively control the virtual reality events.

A new wave of attention towards the use of virtual reality in software visualization systems rose in early 2000s. About a decade ago papers describing the opportunities of virtual reality in the field of software objects representation on the basis of city and landscape metaphors started to emerge [9], [10]. As far as this decade, papers in this field have received a big boost.

The paper [8] describes the prototype of a software visualization system based on a city metaphor. The system employed a VR-headset. The interface was designed based on sign language.

For initial assessment of the system the authors carried out structured interviews, the participants of which had to solve three problems on understanding a program and assess the convenience of using the gestures, as well as indicating their general impression from using the means of virtual reality in understanding the programs essence. At the same time, the same research team published a paper on using physical modelling on the basis of 3D-printing for the purposes of software visualization [7]. Software packages visualization is carried out on the basis of a city metaphor. As a result, users get physical models of a programs objects, which increases their capabilities to perceive and understand the result of their work. The presence of physical models also promotes better rapport among the development process participants. The opportunity of using these models in educational process is presupposed.

A city metaphor is rather popular among software visualization systems using the means of virtual reality. In this respect, two systems with similar names are worth mentioning VR City [28] and CityVR [16].

The VR City system employs a modified city metaphor to represent program systems and related analytical data. Static (metrics) and dynamic (traces) aspects of programs are visualized. Users can observe and interact with objects of the city in an immersive virtual reality environment. A program browsing function is available.

Note that the publication [16] discusses not only the use of a city metaphor in software visualization based on virtual reality, but also the approach of **gamification** of software engineering tasks. This approach presupposes creating tools that provide software engineers with an interface similar to that of computer games. The analysis of software engineers interactions with the CityVR visualization system is carried out. The engineers were excited, felt motivation to work, a certain challenge, experienced immersion into the virtual world, with retaining the system controllability. They spent a significant amount of time on navigation in the virtual world during interaction and while choosing the necessary elements of the program. Users understood that time ran faster than in reality and that is why they were ready to spend more time using software tools to solve the problem of understanding the task.

Gamification in software visualization systems development based on virtual reality is also mentioned in the paper [18]. An environment based on virtual reality is described, which should provide work with the structures of a program code using city metaphors and cosmic metaphors for visualization, navigation and program code data transfer in an interactive mode. Games have been released that have demonstrated the potential of gamification for the purposes of enhancing the understanding of structural dependencies and code modularization. Also worth mentioning are the publications of this research team, devoted to software visualization systems development based on virtual reality [18], [20] (including the paper on visual programming with the use of virtual reality [17]). The paper [19] discusses the prospects of using immersion for software engineers into the program's structures, and paper [20] a virtual flight over software objects within various metaphors (including city metaphors and cosmic metaphors).

Another paper [23] deals with software system visualization based on virtual reality. This tool enables software engineers, project managers or clients to explore the architecture and get a first impression of the dimensions and interdependencies of components. The virtual scenes use simple graphics cubes of various sizes and colors representing software objects, with the demonstration of dependencies in the form of trace lines.

The paper [22] discusses using mixed reality for code checkout. An interactive 3D visualization of the message flow is presupposed, combining traditional visualization techniques with augmented reality based on a VR-headset.

The paper [14] is devoted to the issues of usability assessment of software visualization systems. However, to a certain extent, it could be presented as a development of the CityVR project. The CityAR system was created on the basis of an immersive extended reality environment. Within a city metaphor, buildings are used to represent classes, and their sizes and colors code the software metrics.

The abovementioned paper [17] presents a system for software engineers based on virtual reality. The system should provide visualization of the program code structure based on several metaphors. A reading view is implemented with a real keyboard and mouse, as well as interaction within mixed reality. This provides support for main software tasks without the necessity to exit the virtual reality environment. A system prototype has been realized. Research has been carried out demonstrating the feasibility of these ideas. Empirical estimate results have been collected, demonstrating the systems potential.

The paper [6] describes the tool for visual programming with the use of immersive virtual reality. It is supposed to use the system to create built-in digital packages. The system makes it possible to describe complex objects and their connections, set up complex logical structures, and visualize data flows between the real objects of the developed packages in real time. An imagery typical for applications is used. Initial analysis of experimental results has shown that participants of various levels of qualification can successfully create and maintain programs within the given scenarios.

Worth mentioning is also the paper devoted to 3D software visualization systems performance evaluation [15]. For the purposes of the experiment, 3D visu-

alization based on a city metaphor was realized on a standard computer screen, in a virtual reality environment and with a physical 3D printed model. The participants (who were divided into three groups one group per environment) carried out visualization of software systems of various sizes, solved a number of tasks regarding understanding the results and filled in a questionnaire. The efficiency of visualization was evaluated from the viewpoint of performance, a set of impressions and user experience. Although software engineers who used physical visualization spent the least amount of time on identifying mavericks, the least significant difficulties were observed with system visualization based on standard screens. That said, software engineers who used virtual reality means gained the greatest impressions.

A novel real-world metaphor was presented in [24]. This metaphor based on an island system for visualizing module-based software architectures. The entire software system is represented as an ocean with many islands on it. Also the approach for the exploration of software projects in VR was described.

The paper [25] describes the continuation of previous research and development in the paper [24]. The authors present the visualization of component-based software architectures in Virtual Reality (VR) and Augmented Reality (AR). This visualization also is based on an Island Metaphor.

Even a brief overview shows that the agenda of using virtual reality means in software visualization arouses interest. At the same time, the process is currently on prototype development stage and the evaluation of virtual reality efficiency potential for software visualization. It appears that research and pilot projects should be extended with the use of various visualization metaphors. Performance evaluation should be carried out with the use of psychological techniques, in particular, those relying on the activity theory.

4 The development of prototypes of software visualization systems based on virtual reality environments

Analysis of the examples of software visualization systems based on virtual reality environments shows that despite the 25 years of history a lot of questions remain as to the practical applicability of these environments. All this requires additional research and development of systems with account of specific software tasks. This section discusses projects and implementations of prototypes for software visualization systems based on several metaphors and with potential for using virtual reality environments.

4.1 Software visualization based on a city metaphor

A software visualization project is being developed for the needs of software engineers, programers and testers, with possible use of virtual reality. The use of two forms of representation is presupposed: graphs and the one based on a city metaphor.

Let us consider a prototype of a software visualization system which is based on an extended city metaphor, through the addition of active agents. With the help of agents, one can conduct checkouts, testing and comparative analysis of the code with various types of data, logic and other features, responsible for the quality of the code. And due to the links, the user can monitor the interaction of various pieces of code, determine the type of connection, and have the opportunity to rationalize the logic of this code for smart use of quick or closest to quick connections that increase the speed of algorithm operation.

The system provides interaction with the user by forming the visualization of a chosen project with output of information on all the component units of the code. Visualization based on a city metaphor is supposed to be more convenient for testers, as the user, after identifying the problem, can employ the corresponding visualization of the code based on graph representations. That is why graph representation is used for code research and acquiring information about its structure at the moment, while using a city metaphor makes it possible to get the information for code correction. The possibility of teaching the system by comparing code pieces and searching for approximately equal places is considered. This will make it possible to forecast the influence of code changes on its work.

When developing software controlling the work of real items and mechanisms it is practical to visualize their work using natural graphic. Virtual reality makes it possible to construct presentations within objects, to place various transmitters in a 3d model and to see how the device functions under the program control.

A software visualization system prototype has been developed. Its interface is a menu with an option to choose a software engineer activity type (engineer, tester etc.), with level s of access for different kinds of work and for solving different tasks. After choosing a type of activity the user can choose either a project that they wish to work on, a project that they want to explore, or they can choose to start writing a new project.

It is supposed that other metaphors can also be used in creating a new project; for instance, one of the cosmic metaphors (the heliocentric metaphor described below) can enable visual correction of the structure and logic of the algorithms in the code, for quick realization of the initial version of the software.

If an already existing project has been chosen, then there is opportunity to work in the framework of virtual reality with 3D visualization of the code structure based on a city metaphor, which provides brief and quality description of the work of the algorithms in this project. When choosing a task that is a part of the algorithm logic description, a corresponding visual presentation pops up with a window of the algorithm itself, its description, its versions revision and reset history, as well as various statistical data etc. At the same time, each type of activity presupposes its own data presentation.

In the course of visual world realization in a virtual reality environment a corresponding scene is created, which reflects a city representing the project that the software engineer is planning to work on (see Fig. 2).

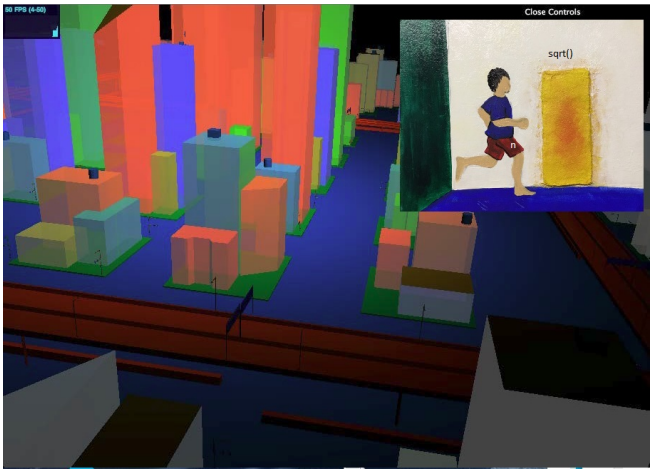

Fig. 2. Presentation of a software project as a city in virtual reality and the inner part of the structure of the code with the figure of the active agent.

The structure of a virtual city corresponds with the structure of a directory tree and file tree of the respective project: a block and a sub-block are a folder and a subfolder respectively, and a building represents a file. In this case, roads represent the borders of the first level of the projects subfolders. The next level of a software project will be represented by building interior. Other objects of a software package can be represented as trees, piping, power stations etc.

In the future, it is planned to conduct research into the possibilities of visual representation of a code structure in virtual reality with enhanced quality of an image and addition of the ability to visualize codes of various programming languages on different platforms, media, to define the best interactive option.

4.2 Visual programming system based on a cosmic metaphor

A project for a programming system based on a visual language built on the principles of object-oriented programming is under consideration. The purpose of this project is to create a medium for visual programming with the possibility to disregard the rules and standards of a specific language, which makes it possible to focus on the problem at hand when the coding. This way, a software engineer will get a better understanding of the structure of a program and minimize the number of related mistakes, such as type discrepancies, syntax interruption etc., by manipulating graphic objects on different levels of abstraction.

A cosmic metaphor in its modern sense, with a heliocentric worldview, is chosen as an idea for a visual programming medium. At the same time, part of the entities of the program is represented as planets, their satellites, rings (like the rings of Saturn), and other elements of outer space (see Fig. 3).

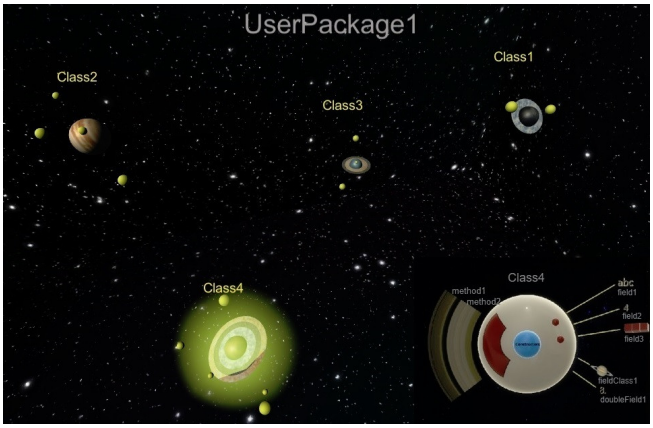

Fig. 3. View of the visual programming environment

It is supposed to exercise interaction with the system by means of devices used in the environments of virtual reality, and a traditional keyboard is used for supplementary tasks, for example, for identifier coding. A task is set to realize the scalability (taking into account various levels of abstraction), to provide navigation along the code and easy transition between the levels of abstraction. The increase of software complexity should not amplify the complexity of graphical representation.

All user classes within this visual programming system are represented as planets. Each planet (class) has two types of views: a free one (traditional view of a planet with its satellites and rings) and an active one (when a class is chosen, it looks like a planets elevation drawing). The active view is a circle with a sector of rings on the left and ordered satellites on the right. The ring sector represents class methods. Each ring is a separate method. Outer rings are public methods; inner rings are private and protected ones. The satellites represent class fields. From top to bottom first go the satellites which are more remote from the planet (public fields), then less remote ones (static methods and fields), which belong to the class itself, and in the center there is a core, which keeps all class constructors within itself. The algorithm (a set of instructions) can be implemented as a context, that is as an area of space with a sequence of expressions that have their own visual representation. Each visual representation can be put in any place of this area. The order of implementation is supposed to be set manually.

In the future, extension of visual software development medium is possible, for instance, by means of introducing graphic analogs of syntactic constructs from Java, so that one could create programs of any complexity. Support for connecting external libraries and classes created outside the system can also be introduced.

4.3 Software visualization based on a geocentric metaphor

A two-dimensional and a three-dimensional option of this metaphor realization are discussed, as well as other possible modifications. Note that the two-dimensional option was suggested as far back as early 90s of the previous century, for transputer networks display [5].

Two prototypes of visualization systems are suggested in the geocentric metaphor: in 2D and 3D versions.

A 3D option of the geocentric metaphor

The metaphor of a geocentric system in 3D space is used for representing the data on supercomputer performance at a certain moment in time. A parallel supercomputer consist of a number of processors. An IT administrator studies the data of the supercomputers work with the aim of increasing its efficiency. Visual representation within this metaphor consists of several layers:

0. In the center of a geocentric system, there is a supercomputers file subsystem.
1. The first sphere, surrounding the center, shows the supercomputers users, whose tasks are being accomplished at a given moment in time.
2. The second sphere contains the tasks being accomplished. The size of the tasks object means the number of supercomputers computation nodes dedicated for the task.

Connections between the layers mean the number of reading and recording operations performed by users tasks on the file subsystem. The thicker the connection, the more such operations is being performed. Thus, a researcher can observe the list of users and their tasks that form the file subsystem load.

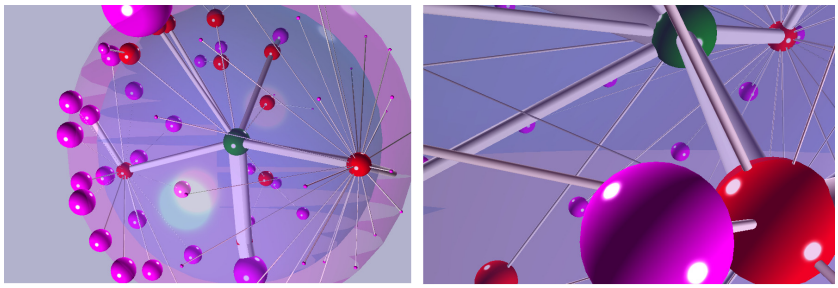

Fig. 4. Left: Visualization of the load on the file system of a supercomputer using geocentric metaphor. Right: A fragment of the state of the supercomputer at some point in time, made from within a three-dimensional geocentric model. A view from within.

An example of the above-described view work process is demonstrated in Fig. 4. In this case, one can see that the main load on the file system (the green sphere in the center) is carried only by several users. Also visible is the texture

of supercomputer use: some users are working on one task, others are working on two or three tasks. However, there are exceptions: a user in the right part of the figure has launched about 30 small tasks.

Users tasks are in turn carried out in supercomputer computational nodes. Thus, a third sphere could be built, where computational nodes of the supercomputer are shown. In this case, the tasks will be shown as planets of a unified size, with the nodes used by them shown on the third sphere, around them. By means of different colors and sizes of celestial bodies one can depict additional data regarding computational processes.

It seems that in the case of virtual reality application the most archaic 3D version of a geocentric model may be more convenient, with the Earth represented as being flat, and celestial bodies located on hemispheres covering the flat Earth. Left and right picture on Fig. 5 illustrate performance data representation, visualized as an archaic 3D geocentric metaphor.

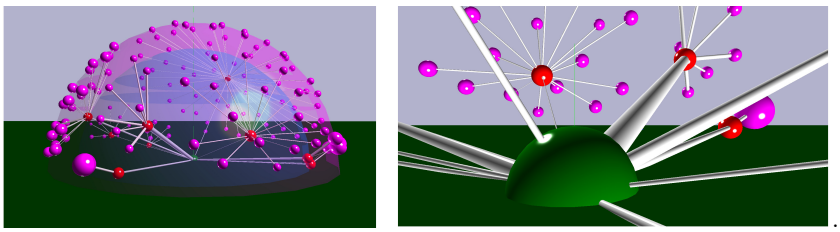

Fig. 5. Left: Visualization of the load on the file system of a supercomputer using an archaic geocentric metaphor. Right: A fragment of the state of the supercomputer at some point in time, made from within a three-dimensional geocentric model. A view from within.

In future, the demonstrativeness of the flat Earth is supposed to increase by means of grid layering. The option with four levels will also be tested. In the case of presenting the supercomputer performance data, the information about users will be placed on the first and the third spheres by the level of their load on the cluster (big and small ones), and the corresponding tasks should be placed on the second and fourth spheres.

Visualization program within the 3D geocentric system metaphor is built on the basis of the Viewlang.ru web library of 3D graphic. This library relies on WebGL standard. It also provides support for WebVR virtual reality technologies. If there is a compatible device (for example, Oculus Rift), the visualization programs user has at their disposal the switch to VR mode button. When pressed, it initiates the graphic transmission onto the VR device. It also reacts to head tilts and users movements. At the same time, the user also has traditional navigation through mouse at their disposal. An example of an image that can be observed by a user in VR mode is shown at the left side of Fig. 4.

4.4 A two-dimensional option of the geocentric metaphor

A two-dimensional option of the geocentric metaphor can be used to represent programs within the paradigm of object-oriented programming. This methodology is based on visualizing a program as a set of objects, each of which is a sample of a certain class, and classes form an inheritance hierarchy.

Based on the assumption that software design using the languages of object-oriented programming is conducted in class-files, the observers viewpoint (the Earth) can be the current work class. The interface system can be represented as a set of satellites of the current planet. In accordance with the perception of stars being far away and immutable objects, third-party dependencies and libraries used in projects can be represented by the metaphor of stars or constellations. Closest celestial bodies-planets represent other classes of the project, their size and appearance depending on various features of the class, its size, used language and file extension.

A system of cascaded placement of the code according to frequency of use or level of connectedness with the current class can serve to interpret numerous transparent celestial spheres. See Fig. 6.

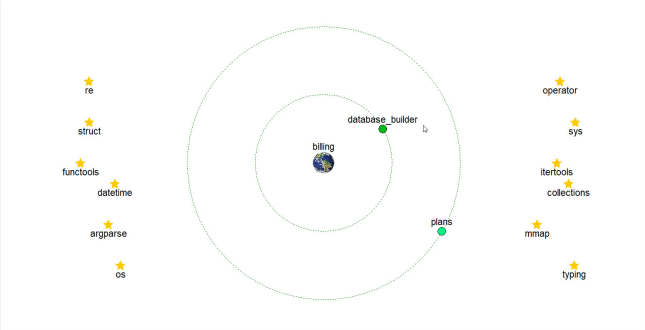

Fig. 6. A two-dimensional geocentric representation of the program within the object-oriented paradigm.

A Python module is chosen on entry and used as a starting point, the Earth. Planets and stars are built based on the imported modules. Planets represent dependencies from the same project, and stars represent external libraries, including standard ones. By choosing a planet one can review the contents of the represented module. Currently a version has been released accepting files with the number of dependencies of up to 20 (out of accuracy and legibility concerns). Expansion of the systems possibilities is planned, in particular, it is intended to add planet assortment based on coupling (how many imported language units are used in the source file) and, possibly, visual representation of the sizes of planet-modules.

Visual representation of object-oriented programs in 3D view is possible for visual programming systems. In this case, the use of virtual reality means is presupposed.

4.5 Visualization based on a starry sky metaphor

Apart from the above mentioned cosmic metaphors, a visualization idea emerged based on a starry sky metaphor. This metaphor was inspired by the images in old engravings.

The example shown in Fig. 7 uses stylized images of stars and constellations. Visualization of up to three levels of data is possible, with the use of sky color and the position of constellations and stars for visualization. At the same time, the stars correspond directly with the objects themselves, constellations with groups or clusters of objects, and the color of the sky can correspond with both super-groups and various features of objects in this part of the sky. For the purpose of visualizing different object features size, shape and color of stars can be used. For increased expressivity (like constellation images in old sky atlases of the ages of Renaissance and Enlightenment) an image corresponding to the given cluster is depicted atop constellations. In our case, the objects of city public transport system are depicted.

This system is suitable for visualization of recurrent or long-term processes by means of moving the objects across the sky with their dynamic development (change of color, position, and size, emergence of new objects and concealing of the old ones).

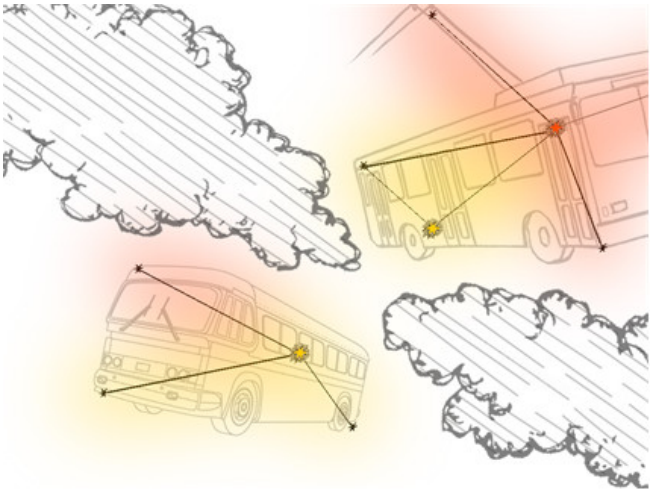

Fig. 7. Example of visualization based on a starry sky metaphor.

For the purposes of filtering objects, images of clouds covering parts of the sky are very suitable. For a more extended and efficient use of this idea one can use virtual reality systems.

5 The human factor in the tasks of software visualization with the use of virtual reality

Software visualization, as any other field of knowledge connected with IT, bumps into the human factor. Software visualization based on virtual reality in particular requires taking the human factor into account and studying it.

Virtual reality as an environment for human activity brings specific states inherent only to it. The first discomfort disturbing normal workflow is cybersickness a distress connected with the conflict of information supplied by various sensory canals: primarily by the optic and the vestibular canals.

One of the cybersickness development factors is the amount of time spent in a virtual environment, that is why work interaction with a VE should take this into account.

Another, more psychological state is the state of presence. The notion of presence is discussed by a number of researchers extensively. The sense of presence is characterized as a basic state of consciousness a conscious feeling of being located in an external world, at the present time. This applies to the physical world, in which our bodies are located, to virtual worlds created through technological mediation, and to a blended mixture of the two: the physical and the virtual. [29]. However, within the framework of this paper, we shall focus on the notion of presence in virtual reality as a special state that is different from presence in the real world, 'presence as the extent to which participants in a VE respond to virtual objects and events as if these were real [11]. Crucial is the fact that a person experiencing the sense of presence perceives the surrounding world in a distorted manner: they forget where they are; they may forget about the people around them if they are not presented in the virtual environment; they may knock out familiar objects from the real world. That is why presence in virtual reality should be considered as a special state that can influence performance. In the case when a virtual environment is used for education, for instance, as a simulator for skill training, one can expect positive influence. In our situation, we are talking about using virtual reality for software visualization. This paper describes system prototypes that employ metaphors, which present abstract data as familiar objects. The engineer faces the challenge of simultaneously operating familiar images and abstract notions. Will presence occur in this situation? How is it going to influence performance? For instance, when using a city metaphor, the actor is supposed to roam the streets, enter buildings and, at the same time, sort out the structure of a software code and data. Can we talk about presence in such an environment, and what exactly is going to happen with this environment? Are we sure it will not be perceived as a city instead of software data presentation?

Speaking about the human factor in software visualization, one cannot but mention the issue of control and navigation. It is most convenient to look to 3D controllers or to use motion capture technology; however, the latter raises questions connected with precision and alignment. This partially depends on the used technologies and programs, but there is still the question of human factor, of distorted perception of distances in a virtual environment. The issue is solved to a degree, as is shown in the paper [26]: familiar size is taken into account when reaching to a virtual object. Thus, a deviating depth perception might be less a problem for 3D touch interaction when reaching to objects with known sizes as compared to abstract objects. That is why an engineer should familiar objects with known size when possible. However, when organizing interaction it is not always possible to use metaphoric objects with a known size. In this case, interaction should not require precision in reaching to the objects of the environment.

6 Conclusion

The development of prototypes of systems based on virtual reality is just the first stage of research. These systems can help bring light upon the issues that could not be resolved in the available publications. It is necessary to search for the criteria of visualization metaphors and approaches to developing views. It is not clear as to how effective the virtual reality environments are in solving various software visualization tasks. Numerous questions remain concerning the adaptation of such systems with regard to user peculiarities, including individual perception limitations and restrains on the time spent in virtual reality environments. The gamification idea is surely extremely interesting, but it cannot always be employed in major development. New research and pilot projects are required in the field of virtual reality application for software visualization systems. Software design for real-life tasks requires specialization, and even personalization of visual systems. Naturally, it is necessary to extend the psychological knowledge regarding interaction with virtual reality and the processes occurring in the course of a professionals activity while working on software visualization systems.

References

1. Averbukh, V.L.: Semiotic Analysis of Computer Visualization, chap. 6, pp. 97–133. InTech, Rijeka, Croatia (2017)
2. Averbukh, V.: Visualization metaphors. Program and Computer Softw. **27**, 227–237 (2001)
3. Averbukh, V., Bakhterev, M., Baydalin, A., Gorbashevskiy, D., D.R., I., A.U., K., Nebogatikova, P., Popova, A., Vasev, P.: Searching and Analysis of Interface and Visualization Metaphors, chap. 3, pp. 49–84. InTech, Vienna (2008)
4. Averbukh, V., Baydalin, A., Ismagilov, D., Kazantsev, A., Timoshpolskiy, S.: Utilizing 3d visualization methophors. In: Proceedings of 14th International Conference on Computer Graphics and Vision, Graphicon 2004. pp. 295–298. Moscow, Russia (2004)

5. Averbukh, V., Konovalov, A., Balakin, V., Gomon, M., Sleptsov, P.: The method of visual display of transputer networks. In: Transputer systems and their application: Tez. konf., Domodedovo. 3–6 okt. 1994. Institute of Applied Mathematics of the Russian Academy of Sciences, Moscow (1994)
6. Ens, B., Anderson, F., Grossman, T., Annett, M., Irani, P., Fitzmaurice, G.: Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments. In: Proceedings of the 43rd Graphics Interface Conference GI'17. pp. 156–162 (2017)
7. Fittkau, F., Koppenhagen, E., Hasselbring, W.: Research perspective on supporting software engineering via physical 3d models. In: 2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT). pp. 125–129 (2015)
8. Fittkau, F., Krause, A., Hasselbring, W.: Exploring software cities in virtual reality. In: 2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT). pp. 130–134 (2015)
9. Glander, T., Dôllner, J.: Abstract representations for interactive visualization of virtual 3d city models. Computers. Environment and Urban Systems **33**(5), 375–387 (2009)
10. Glander, T., Dôllner, J.: Automated cell-based generalization of virtual 3d city models with dynamic landmark highlighting. In: The 11th ICA workshop on generalization and multiple representation, June 20–21 , Montpellier, France. The International Cartographic Association (2018)
11. Khanna, P., Yu, I., Mortensen, J., Slater, M.: Presence in response to dynamic visual realism: A preliminary report of an experiment study. In: Proceedings of the ACM symposium on Virtual reality software and technology. pp. 364–367. ACM (2006)
12. Maletic, J., Leigh, J., Marcus, A.: Visualizing software in an immersive virtual reality environment. In: Proceeding of the ICSE'01 Workshop on Software Visualization. pp. 49–54. Toronto, Canada (2001)
13. Maletic, J., Marcus, A., Collard, M.: A task oriented view of software visualization. In: International Workshop on Visualizing Software for Understanding and Analysis. pp. 32–40 (2002)
14. Merino, L., Bergel, A., Nierstrasz, O.: Overcoming issues of 3d software visualization through immersive augmented reality. In: 2018 IEEE Working Conference on Software Visualization (VISSOFT). pp. 54–64 (2018)
15. Merino, L., Fuchs, J., Blumenschein, M., Anslow, C., Ghafari, M., Nierstrasz, O., Behrisch, M., Keim, D.: On the impact of the medium in the effectiveness of 3d software visualizations. In: Proceedings on 2017 IEEE Working Conference on Software Visualization (VISSOFT). pp. 11–21 (2017)
16. Merino, L., Ghafari, M., Anslow, C., Nierstrasz, O.: Cityvr: Gameful software visualization. In: IEEE International Conference on Software Maintenance and Evolution (ICSME TD Track). pp. 633–637 (2017)
17. Oberhauser, R.: Immersive coding: A virtual and mixed reality environment for programmers. In: Proceedings of The Twelfth International Conference on Software Engineering Advances (ICSEA 2017). pp. 250–255 (2017)
18. Oberhauser, R., Carsten, L.: Gamified virtual reality for program code structure comprehension. The International Journal of Virtual Reality **17**(02), 79–88 (2017)
19. Oberhauser, R., Lecon, C.: Immersed in software structures: A virtual reality approach. In: ACHI 2017. The Tenth International Conference on Advances in Computer-Human Interactions. pp. 181–186 (2017)

20. Oberhauser, R., Lecon, C.: Virtual reality flythrough of program code structures. In: Proc. of the 19th ACM Virtual Reality International Conference (VRIC 2017). ACM (2017), article No. 10
21. Reed, D., Scullin, W., Tavera, L., Shields, K., Elford, C.: Virtual reality and parallel systems performance analysis. IEEE Computer **28**(11), 57–67 (1995)
22. Reipschläger, P., Gumhold, S., Ozkan, B., Majumdar, R., Mathur, A., Dachselt, R.: Debugar: Mixed dimensional displays for immersive debugging of distributed systems. In: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems (2018), paper No.: LBW117
23. Schreiber, A., Bruggemann, M.: Interactive visualization of software components with virtual reality headsets. In: 2017 IEEE Working Conference on Software Visualization (VISSOFT). pp. 119–123 (2017)
24. Schreiber, A., Misiak, M.: Visualizing software architectures in virtual reality with an island metaphor. In: Chen, J., Fragomeni, G. (eds.) Virtual, Augmented and Mixed Reality: Interaction, Navigation, Visualization, Embodiment, and Simulation. VAMR 2018. Lecture Notes in Computer Science, vol. 10909. Springer (2018)
25. Schreiber, A., Misiak, M., Seipel, P., Baranowski, A., Nafeie, L.: Visualization of software architectures in virtual reality and augmented reality. In: IEEE Aerospace Conference Proceedings. IEEE Aerospace Conference 2019, March 02–09 2019. Big Sky Montana, USA (2019)
26. Schubert, R.S., Müller, M., Pannasch, S., Helmert, J.R.: Depth information from binocular disparity and familiar size is combined when reaching towards virtual objects. In: In Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology. pp. 233–236 (2016)
27. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: Proceedings of the IEEE Conference on Visual Languages, September 3–6, 1996. pp. 336–343 (1996)
28. Vincur, J., Navrat, P., Polasek, I.: Vr city: Software analysis in virtual reality environment. In: 2017 IEEE International Conference on Software Quality. Reliability and Security Companion. pp. 509–516 (2017)
29. Waterworth, J., Riva, G.: Feeling Present in the Physical World and in Computer-Mediated Environments. Palgrave Macmillan, London (2014)