Manipulating Function-Based Objects with Interactive Collision Risk Models

Sergey Vyatkin[1][0000-0002-1591-3588], Olexandr Romanyuk[2][0000-0002-2245-3364], Sergii Pavlov[2][0000-0002-0051-5560], Pavlo Mykhaylov[3][0000-0001-5861-5970], Roman Chekhmestruk[4][0000-0002-5362-8796], Oksana Vodzinska[5][0000-0002-1246-7156], Olena Prozor[2][0000-0003-1454-8352] and Volodymyr Lytvynenko[6][0000-0002-1536-5542]

[1]Institute of Automation and Electrometry, Ak. Koptyuga, 1, 630090, Novosibirsk, Russia,
sivser@mail.ru

[2]Vinnytsia National Technical University, Vinnytsia, Khmelnytske shose 95, Ukraine
rom8591@gmail.com, psv@vntu.edu.ua, el.przr@gmail.com,
maya.kovalchuk@gmail.com

[3]CEO 3D GNERATION GmbH, Viktoriastraße 15, 44137, Dortmund, Germany
pm@3dgeneration

[4]3D GENERATION UA, 21021, Pirogova Str. 37. 21021Vinnitsa, Ukraine
Rc.ua@3dgeneration.com

[5]Kyiv National University of Technologies and Design, Ukraine
oksiiv@ukr.net

[6]Kherson National Technical University, Ukraine
immun56@gmail.com

Abstract. A method for interactive manipulation of function-based objects taking into account the physical properties of the environment is proposed. To ensure interactivity, the efficient acceleration methods for function evaluation have been proposed. This method can be used to simulate the movement of solids in the field of gravity, taking into account the definition of collisions. A model describing the collision of solids is also proposed. A dynamically loaded program has been developed and implemented that allows you to interactively manage the scene. Using the program the motion of solid bodies in the field of gravity is modeled, taking into account collisions. The following features are implemented: using different three-dimensional scenes; support for various types of control devices; work with tracks; work with scenarios. The proposed method can be used in computer science, information technologies, computational simulation and the risk field. For example, the method can be used for ship collision risk models and for unmanned surface vehicles in which obstacles (i.e., collision risks) are determined through encounter situations.

Keywords: function-based objects, interactive manipulation, movement of solids, collision detection, computational simulation, risk-informed systems.

1 Introduction

Real-time computer graphics, focused on the visualization of three-dimensional scenes, has achieved significant success to date. It is widely used from visualization systems of training complexes (aviation, space, marine, automobile, etc.) to computer games. Real-time visualization, which provides a high enough realism of the depicted objects, requires a lot of computing power. Systems designed to solve this problem are based on graphics processing units (GPUs) [1,2,3]. Virtual objects created by the designer are placed in the virtual scene, which should move, collide and rotate in the closest way to reality [4,5,6]. Several representations of geometric objects are currently used in computer graphics. Each of the objects, according to its properties, is used in different fields, beginning from 3-D simulation and CAD systems up to real-time visualization systems. With the development of ship collision risk models, intelligent optimization methods have been investigated to achieve real-time and reliable ship collision avoidance, such as neural networks, fuzzy mathematics and reinforcement learning. Meanwhile, fuzzy mathematics can also be applied to fuzzy collision risk classification and fuzzy inference. The performance of fuzzy mathematics depends on a preset membership function, which requires more a priori knowledge. An autonomous motion planning algorithms for unmanned surface vehicles in which obstacles (i.e., collision risks) are determined through three basic encounter situations important also. The functional representation describes most accurately the object geometry and has the smallest size of the required data. Procedures of functional representation demonstrate compact and flexible representation of surfaces and objects that are the results of logical operations on volumes [7,8,9].

One of the tasks in developing real-time visualization systems is interactive control of the scene or its individual parts. A scene is a set of objects and their characteristics that are necessary for visualization. There are two main ways to control the behavior of scene objects: using tracks, script commands. The use of tracks allows you to create scenes with complex object behavior. This is a very common method, but it does not fully implement interactivity. Although it is possible to manipulate tracks (set the desired frame, play the frame range in the forward or reverse direction), tracks are a prepared set of frames and cannot be changed while working. A set of script commands for controlling the position of an object allows you to set the following characteristics: position in the coordinate system (x, y, z), rotation around axes $(\alpha_x, \alpha_y, \alpha_z)$, and scaling along each of the axes (s_x, s_y, s_z). It is also possible to set the speed of changes in all of the above characteristics.

When using commands, there is a problem related to the fact that the scene may go into an invalid state, for example, an object collision or a scene split. In other words, there is no registration and processing of exceptional situations that occur when the integrity of the scene state is violated, such as a collision or an object leaving a valid area. When using devices that are most suitable for interactive management, it is difficult to link their state and the behavior of the scene. We can associate a value that characterizes the state of a control device with a given characteristic (or set of characteristics, but they will all change according to the same law) of a scene or object, but this characteristic has a simple (linear) dependence on the state of the device.

This work is devoted to the development of a method for interactive control of three-dimensional scenes, taking into account the physical properties of the environment [10,11,12]. The control of the position of objects is considered: offsetting, rotation, scaling, etc.

2 Problem statement

It was necessary to develop and implement a method that allows you to interactively manipulate the 3D scene and its individual objects. It should include managing the position of objects and working with tracks.

Requirements for the method [13,14,15]:

1. Real-time operation;
2. Registration and handling of exceptional situations;
3. Accounting for the state of control devices when determining the behavior of the scene.

3 Interactive system modeling

We have developed an algorithms and a set of *C++* classes for system modeling: recursive multilevel ray casting [9] for scenes containing functionally defined objects (including OpenGL color/depth buffers compatibility); *C++* classes for functionally defined objects representation; *C++* classes for rendering of functionally defined objects; *C++* classes interface, these classes provided to make the whole system to be easily extended to incorporate new algorithms and features. Thus, the resulting is designed as collection of classes (VxFramework) to facilitate the development of system modeling applications [16,17,18].

The modules are subdivided other tasks of the projects into independent parts that use the classes of *VXFramework* and are integrated into the system through inheritance of interface classes. The VxDll module is responsible for Base classes for rendering and objects representation. The VxSceneBuilder.dll module is responsible for scene parsing/saving. An interactive system modeling stuff is grouped in *VxManipulator* class.

4 Interactive scene management method

The behavior of the scene and its individual parts must be subject to certain rules or laws. A virtual world is created (environment) with its own rules, and the behavior of objects in the environment obeys these rules. The rules for scene behavior may differ for different tasks, and are selected depending on the requirements of a particular task.

Requirements for the rules are:

1. Description of all possible states;
2. Registering the occurrence of invalid states;

3. Description of the scene behavior in exceptional situations;
4. Determining the reaction of the scene to the commands of the control device [19,20,21].

We introduce an additional set of characteristics (both for the entire scene and for individual objects) necessary to describe the behavior of the scene within the specified rules. In geFneral, this set may not overlap with the set of geometric characteristics used for visualization.

The state of the scene must always obey the laws of the environment, i.e. $\overline{H}(\overline{E}) = 0$, where $\overline{E}$ is the vector of characteristics of the state of the scene, and $\overline{H}$ is a system of equations describing the laws of the environment. From the characteristics of the scene state, we must be able to find geometric characteristics. In other words, there must be a vector function $\overline{F}$ such that $\overline{X} = \overline{F}(\overline{U})$, where $\overline{X}$ is the vector of geometric characteristics of the scene.

Characteristics of the state of the scene can be passive and active. Active characteristics are those whose value is determined by means of control devices; passive - characteristics that completely obey the laws of the environment. The active vector of characteristics denoted by $\overline{K}$, while the passive - $\overline{U}$, then $\begin{cases} \overline{H}(\overline{U},\overline{K}) = 0 \\ \overline{X} = \overline{F}(\overline{U},\overline{K}) \end{cases}$. Moreover, you can choose the vectors $\overline{U}$ and $\overline{K}$ so that the vector function $\overline{F}$ will depend only on the vector $\overline{U}$, i.e. $\begin{cases} \overline{H}(\overline{U},\overline{K}) = 0 \\ \overline{X} = \overline{F}(\overline{U},\overline{K}) \end{cases}$. In the case of modeling, when the characteristics change not continuously over time, but discretely, you can write a) $\overline{U}_{n+1} = \overline{H}(\overline{U}_n, \overline{K}_n)$, b) $\overline{X}_{n+1} = \overline{F}(\overline{U}_{n+1})$.

In this case, $\overline{H}$ it does not denote a system of equations, but a vector-function of the transition to the next state.

At the initialization stage (see figure 1), we set the initial state of the scene – the vector $\overline{U}_0$. During the survey of control devices, we determine the vector of active characteristics $\overline{K}_n$. Then find a new scene state $\overline{U}_{n+1}$, and then register the existence of the exception, if it is present, then call processing exceptional situations, if not, go to step calculate the geometric characteristics $\overline{X}_{n+1}$.

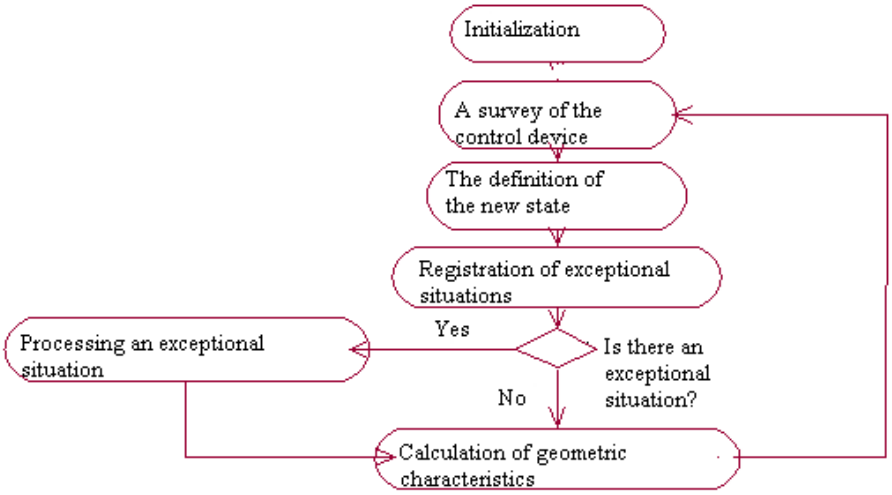

Fig. 1. How the method works

5 Motion of a solid body

The laws of motion of a solid body were chosen by us as an example of rules describing the behavior of a scene. These laws describe the behavior of objects quite realistically. There is a gravitational field in the medium, and collisions of physical bodies are registered and calculated.

The motion of a free solid is described by the following differential equations [1,22,23]: a) $\frac{d}{dt}(mV_C) = F^{ext}$, b) $\frac{dL_C}{dt} = M_C^{ext}$, where m is body mass, V_C is the speed of its center of mass C, F_C is the main vector of external forces applied to the body, M_C^{ext} is the main moment of external forces relative to the point C , and L_C is the angular momentum of the body relative to the same point C.

In the case of modeling, it is more convenient to use these equations in integral forma) $\delta(mV_C) = \int_{\Delta t} F^{ext} dt$, b) $\delta L_C = \int_{\Delta t} M_C^{ext} dt$, where $\Delta t = t_{n+1} - t_n$ is the time elapsed between two neighboring iterations.

To describe the state of the scene, new characteristics are introduced for each scene object: mass, moment of inertia, momentum, and moment of momentum.

6 Collision detection

An effective solution to this problem is described in [8], which describes this process quite reliably. The advantages of the proposed solution are:

– Close to reality results;
– The ability to consider the impact as an instantaneous process;
– Collision parameters are easy to understand and are clearly demonstrated during visualization.

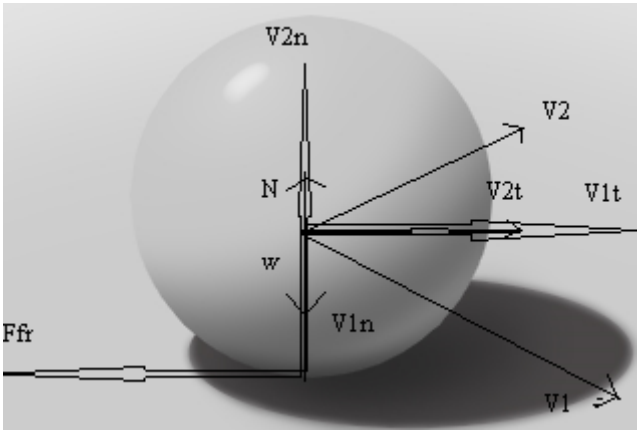

Fig. 2. Determining the collision of a ball with a plane

It is assumed that the deformation of objects is much smaller than their linear dimensions, and the interaction time is much less, than the time required for objects to travel a distance comparable to their linear dimensions.

For each pair of objects there is a parameter α lying in the interval from 0 to 1, characterizing the so-called "elasticity" of the collision. It is equal to the ratio of the projection module end speed of the normal to the point of impact to the module of the projection of the initial velocity to the same normal (see Fig. 2): $|V_{2n}| = \alpha |V_{1n}|$, $\alpha \in [0;1]$, where V_{1n} is the normal component of the initial velocity vector, and V_{2n} is the normal component of the final velocity vector. Thus, the effect Act_n of the impact force N is equal to $Act_n = \int N dt = \delta p_n$. Since the direction of the impact force does not change during the impact, the normal value of the action is $|Act_n| = \left|\int N dt\right| = \int |N| dt = |\delta p_n| = (1+\alpha) m |V_{1n}|$. We assume that the coefficient of friction μ remains unchanged and the fricAtion force always acts in the same direction, then the action of the friction force is equal to $|Act_t| = \int |F_{fr}| dt = \int \mu |N| dt = \mu |Act_n|$, $\mu \in [0;1]$. The action of the friction force is directed against the sliding speed $V_s = V_{1t} + \varpi_1 \times r$, where ϖ_1 is the angular velocity vector of the ball before it hits its target, and may not exceed the actions necessary to stop the slide i.e. $\left(V_{1t} + \frac{Act_{t\max}}{m}\right) + \left(\varpi_1 + \frac{r \times Act_{t\max}}{J}\right) \times r = 0$,

where Act_{tmax} is the maximum action of the friction force, J is the moment of inertia of the ball relative to its center. The maximum possible action of the friction force is equal to $\left|Act_{t\max}\right| = \frac{\gamma}{\gamma+1} m\left|V_S\right|$, where $\gamma = \frac{J}{mr^2}$. Accordingly, the value of the action of the friction force is equal to $\left|Act_t\right| = \min\begin{cases} \nu\left|Act_n\right| \\ \left|Act_{t\max}\right| \end{cases}$.

Let's record the changes in the momentum $\delta(mV_C)$ and moment of momentum δL_C for the ball $\begin{cases} \delta(mV_C) = Act_n + Act_t \\ \delta L_C = r \times Act_t \end{cases}$.

Thus, we know the values of all the values necessary for further calculations.

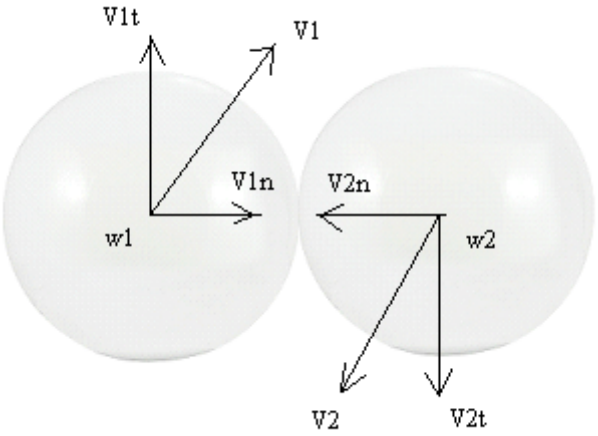

Fig. 3. Collision detection of two balls

When two balls collide (see Fig. 3), it can be assumed that the balls collide with a plane with velocity $u = u_n + u_t$: a) $u_n = \frac{m_1 v_1 + m_2 v_2}{m_1 + m_2}$,

b) $u_t = \frac{\frac{\gamma_1}{\gamma_1+1} m_1 (V_1 + \varpi_1 \times r_1) + \frac{\gamma_2}{\gamma_2+1} m_2 (V_2 + \varpi_2 \times r_2)}{\frac{\gamma_1}{\gamma_1+1} m_1 + \frac{\gamma_2}{\gamma_2+1} m_2}$.

Where the normal component u_n is from the condition of equality of the impact force action, and the tangential component is from the condition of equality of the maximum friction force action, for each ball:

a)m $m_1(1+\alpha)(V_{1n}-u_n)=-m_2(1+\alpha)(V_{2n}-u_n)$,

b) $\frac{\gamma_1}{\gamma_1+1}m_2(V_{1t}-u_t+\varpi_1\times r_1)=-\frac{\gamma_2}{\gamma_2+1}m_2(V_{2t}-u_t+\varpi_2\times r_2)$.

To describe collisions, the coefficient α and coefficient of friction μ are set for each pair of objects.

7 Application for scene control

A dynamically loaded module was written for the Interactive System for Modeling, Animation and Rendering of Functionally Defined Objects [8]. The module expanded the capabilities of the Interactive System, which allowed you to interactively manage the scene in real time, taking into account the physical properties of the environment. Objects in the scene obey the laws of motion of a solid body. The module uses a system of script commands to control the behavior of scene objects.

The basic requirements for the module:

1. The possibility of using different scenes;
2. Support for various types of control devices;
3. Ability to execute scripts in case of exceptional situations;
4. Ability to work with tracks.

The module is implemented in the C++ programming language. This language is chosen to achieve the greatest compatibility with the Interactive System for Modeling, Animating and Rendering of Functionally Defined Objects. Another advantage of this language is its object-oriented approach. The module was developed with the help of an integrated programming environment Microsoft Visual Studio 6.

The module consists of a counting block, a database, a control device handler, and a synchronization block (see figure 4).

The database stores the physical characteristics of the scene and objects necessary to describe the behavior: mass, moment of inertia, momentum, moment of momentum, coefficient and coefficient of friction; of these, the momentum and moment of momentum are calculated, and the rest are constant characteristics of the object.

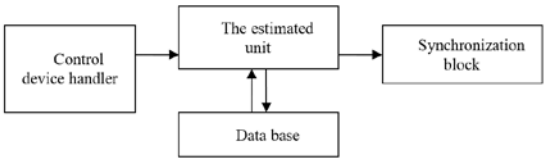

Fig. 4. Structure of the scene control module

The database also stores some geometric characteristics of objects that are used for calculations. In addition, the database contains information about tracks: name, key frames; service information: associations of objects described in the module with

scene objects; connections with control devices and connections with scenarios that are triggered when exceptional situations occur.

The control device handler checks the state of the desired devices and determines the active characteristics of the scene. When working with the keyboard, the device is directly queried, while external commands are processed when working with the joystick. The estimated unit is responsible for calculating the characteristics of the scene. It also registers exceptional situations and processes them. The sync block is responsible for updating the 3D scene.

8 Results of work

A dynamically loaded module has been developed and implemented that allows you to interactively manage the scene. The module simulates the motion of solid bodies in the field of gravity, taking into account collisions. The following features are implemented in the module: using different three-dimensional scenes; support for various types of control devices; work with tracks; work with scenarios.

During initialization, the scene, all its objects, and their initial position are described using script commands. The first stage is creating a three-dimensional scene. At the second stage, the initialization script for the module is enabled. In this scenario, you must: describe the scene and the necessary objects; specify additional characteristics; describe tracks, if any; specify scenarios that will be executed when collisions occur; specify the object on the surface of which information about the state of the scene will be placed. When describing a scene, the module passes the initial state of objects (initial position, size), their names, and the area where objects can move. As well as additional features: field of gravity, mass of objects, moments of inertia, coefficients describing collisions.

Initialization is necessary to ensure independence from a specific scene. At this point, a model or representation of the scene is created, and the module works with it, not with the scene itself. When configuring the module, parameters such as gravity, coefficients for calculating collisions, tracks, and scenarios for calling in case of exceptional situations are set. This setting allows you to change the rules for the behavior of objects during operation. At the configuration stage, active objects are linked to management devices.

The following operations are performed in the module:

1. Processing control devices.
2. Changing the speed and position values of objects with active characteristics. Checking the validity of the state of these objects and, if necessary, correcting them.
3. Calculation of speed and position for other objects.
4. Registration of collisions, checking the validity of the state of passive objects.
5. Correction of the state of passive objects occurs when exceptional situations occur.
6. Syncing the scene.
7. Working with tracks.
8. Sending event alerts.

After calculating the new location of objects in the module, you must synchronize the positions of objects in the scene.

9 Conclusion

The method of interactive control of the scene behavior is described. The work has a wide application. The proposed algorithm for controlling the behavior of a three-dimensional scene can be useful for solving many problems of interactive real-time control for three-dimensional computer graphics, motion planning tasks for unmanned surface vehicles and the development of ship collision risk models. The developed model for determining collisions of solid bodies can be used both for modeling this phenomenon in other systems, and as a theoretical description of collisions in the study of physics. The developed scene control module can be used independently to demonstrate the laws of kinematics and dynamics. It can also serve as a basis for further expanding the capabilities of interactive 3D object management.

References

1. Goldstein, H., Poole, C.P., Jr. Safko, J. L.: Classical Mechanics (third Ed.). San Francisco, CA: Addison Wesley. (2002) ISBN 0-201-65702-3.
2. Dvorak, R., Freistetter, F.: "§ 3.2 Lagrange equations of the first kind". Chaos and stability in planetary systems. Birkhäuser. p. 24. (2005) ISBN 3-540-28208-4.
3. Haken, H.: Information and self-organization (3rd Ed.). Springer. p. 61. (2006) ISBN 3-540-33021-6.
4. Shabana, A.A.: Computational continuum mechanics. Cambridge University Press. pp. 118–119. (2008) ISBN 0-521-88569-8.
5. Padmanabhan, T.: "§2.3.2 Motion in a rotating frame". Theoretical Astrophysics: Astrophysical processes (3rd Ed.). Cambridge University Press. p. 48. (2000) ISBN 0-521-56632-0.
6. Gans, R.F.: Engineering Dynamics: From the Lagrangian to Simulation. New York: Springer. (2013) ISBN 978-1-4614-3929-5.
7. Gannon, T.: Moonshine beyond the monster: the bridge connecting algebra, modular forms and physics. Cambridge University Press. p. 267. (2006) ISBN 0-521-83531-3.
8. Vyatkin S.I., Romanyuk O.V.: Collision detection of solid objects, Days of science in DonNTU: Proceedings of the VI International conference "Modeling and computer graphics" (Krasnoarmeysk, Ukraine, may 26-29, 2015). DonNTU, 2015. pp. 113-124.
9. Vyatkin S.I.: An Interactive System for Modeling, Animating and Rendering of Functionally Defined Objects, American Journal of Computer Science and Engineering Survey, 2014, Vol. 2, iss. 3. pp. 102–108.
10. Wojcik, W., Kotyra, A., Golec, T. et al.: Vision based monitoring of coal flames Source: Przeglad Elektrotechniczny Volume: 84 Issue: 3 pp.: 241-243 Published: 2008
11. Smolarz, A., Wojcik, W., Ballester, J. et al.: Fuzzy controller for a lean premixed burner: PRZEGLAD ELEKTROTECHNICZNY Volume: 86 Issue: 7 pp.: 287-289 Published: 2010.

12. Kaczmarek, C., Wojcik, W. et al.: Measurement of pressure sensitivity of modal birefringence of birefringent optical fibers using a Sagnac interferometer: Optica Applicata Volume: 45 Issue: 1 pp.: 5-14 Published: 2015.
13. Wojcik, W.: Application of fibre-optic flame monitoring systems to diagnostics of combustion process in power boilers: Bulletin of the Polish Academy of Sciences-Technical Sciences Volume: 56 Issue: 2 pp.: 177-195 Published: JUN 2008.
14. Rovira, R.H., Pavlov, S.V., Kaminski, Bayas M.M.: Methods of Processing Video Polarimetry Information Based on Least-Squares and Fourier Analysis // RH Rovira, SV Pavlov, OS Kaminski, MM Bayas - Middle-East Journal of Scientific Research, T. 16 (9), 1201-1204 2013. – pp.1201-1204. DOI: 10.5829/idosi.mejsr.2013.16.09.12002
15. Zabolotna, N. I., Pavlov, S.V., Ushenko, A.G., Sobko O.V., and Savich V.O.: Multivariate system of polarization tomography of biological crystals birefringence networks, *Proc. SPIE* 9166, Biosensing and Nanomedicine VII, 916615 (August 27, 2014); DOI:10.1117/12.2061105
16. Wójcik, W., Smolarz, A.: Information Technology in Medical Diagnostics . London, July 11, 2017 by Taylor &Francis Group CRC Press Reference - 210 Pages. ISBN 9781138299290
17. Vassilenko, S., Valtchev, Teixeira J.P., Pavlov, S.: Energy harvesting: an interesting topic for education programs in engineering specialities / «Internet, Education, Science" (IES-2016) – 2016. – pp. 149-156.
18. Kvyetnyy, R., Bunyak, Y., Sofina, O., and etc.: Blur recognition using second fundamental form of image surface, Proc. SPIE 9816, Optical Fibers and Their Applications 2015, 98161A (17 December 2015). DOI: 10.1117/12.2229103
19. Kvyetnyy, R.N., Romanyuk, O.N., Titarchuk, E.O., and etc.: Usage of the hybrid encryption in a cloud instant messages exchange system, Proc. SPIE 10031, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2016, 100314S (28 September 2016). DOI: 10.1117/12.2249190.
20. Kvyetnyy, R., Sofina, O., Orlyk, P., Utreras, A.J., Wójcik, W., and etc.: Improving the quality perception of digital images using modified method of the eye aberration correction", Proc. SPIE 10031, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2016, 1003113 (28 September 2016). DOI: 10.1117/12.2249164
21. Babichev, S., Lytvynenko, S., Osypenko, V., Korobchynskyi, M., and Voronenko, M.: Comparison Analysis of Biclustering Algorithms With the Use of Artificial Data and Gene Expression Profiles 2018 IEEE 38-International conference on Electronics and nanotechnology, Kyiv,April 24-26, 2018 – pp. 292-298. IEEE Catalog Number: CFP1805U-USB. DOI: 10.1109/ELNANO.2018.8477439
22. Fefelov, A., Lytvynenko, V., Babichev, S., Osypenko, V., and Voronenko, M.: Reconstruction of the gene regulatory network by hybrid algorithm of clonal selection and trigonometric differential evolution, 2018 IEEE 38-International conference on Electronics and nanotechnology, Kyiv, April 24-26, 2018– pp. 298-305. DOI: 10.1109/ELNANO. 2018.8477436
23. Lytvynenko, V., Lurie, I., Krejci, J., Voronenko, M., Savina,N., Ali Taif. M.: Two Step Density-Based Object-Inductive Clustering Algorithm. Proceedings of the Second International Workshop «Modern Machine Learning Technologies and Data Science» (MoMLeT&DS-2019), Shatsk, Ukraine, June 2-4, 2019– pp. 117-136.