PROBLEMY MECHATRONIKI
UZBROJENIE, LOTNICTWO, INŻYNIERIA BEZPIECZEŃSTWA

PROBLEMS OF MECHATRONICS
ARMAMENT, AVIATION, SAFETY ENGINEERING

ISSN 2081-5891; E-ISSN 2720-5266 **https://promechjournal.pl/**

Research paper

Matlab Class for Rapid Visualization of Missile-Target Engagement Scenarios

Witold BUŻANTOWICZ (witold.buzantowicz@wat.edu.pl)

ORCID: https://orcid.org/0000-0002-4737-4857

Military University of Technology,
2 Sylwestra Kaliskiego Str., 00-908 Warsaw, Poland

Received: November 14, 2022 / Revised: January 25, 2023 / Accepted: January 25, 2023 / Published: December 31, 2023.

2023, 14 (4), 23-36; https://doi.org/10.5604/01.3001.0054.1646

Cite: Chicago Style

Bużantowicz, Witold. 2023. "Matlab Class for Rapid Visualization of Missile-Target Engagement Scenarios" *Probl. Mechatronics. Armament Aviat. Saf. Eng.* 14 (4) : 23-36. https://doi.org/10.5604/01.3001.0054.1646

Abstract. In this paper, the authorial Matlab Battlespace Visualization Class (MBVC) is proposed to plan, design, and compose the preliminary version of the desired computer animation, which can then be used both as a background for the discussion of the final composition and as an individual project for public presentation, providing a visual reference for missile-target engagement simulations. As a design tool, the MBVC allows us the combination of the advanced mathematical models of missile guidance loop elements (implemented in the discrete form as program scripts) with the capabilities offered by open, powerful, and flexible 3D graphical engine. As a visualization tool, the MBVC allows the designers to present their research findings in both an illustrative and informative way, providing a visual reference for the computer simulation. The class is freely available for scientific, engineering or educational use.
Keywords: 3D visualization, conceptual design, missile, aerial target, trajectory

1. INTRODUCTION

In the early 21st century, the requirements of anti-aircraft missiles changed significantly. Along with the development of modern air-raid methods, the existing tasks, which mainly involved destroying aircraft and helicopters, were extended to include the neutralization of cruise and aeroballistic missiles, as well as of unmanned aerial vehicles [1-3]. Modern technical solutions arm aerial targets with a variety of effective responses to threats while in the air, e.g., anti-missile maneuvering or use of various types of jamming and traps [4]. Hence, it is becoming increasingly important to seek new technical solutions to ensure highly effective anti-aircraft missile guidance in varying conditions and scenarios. However, experimental testing, necessary for the collection of reliable results, is extremely expensive and it involves a range of organizational and logistical challenges. It is not economically justified at the initial stages when formulating assumptions or verifying the correctness of a design concept. Simulation methods therefore play an important role.

The requirement for interdisciplinary collaboration from the science and technology fields, and the need to adapt to the extreme conditions of air combat, has resulted in anti-aircraft missiles becoming highly complex weapon systems. This complexity is reflected in the process of building mathematical simulation models to assess their effectiveness. The nonstationary and nonlinear character of the phenomena under consideration makes it impossible to find general solutions. In other words, high-credibility simulation models are built using mathematical modelling of physical phenomena, taking into account technical conditions specific to the considered issues, while solutions are sought using numerical methods.

When assumptions are formulated for new (future) weaponry variants, it is very useful at the analytical and conceptual stage to have a tool that enables an efficient, reliable, and multi-faceted evaluation of the proposed solutions.

Furthermore, what is required is a tool that provides great flexibility in introducing the discussed changes, but also – very importantly – one that can present the developed variants of the concept to a wider audience in a visually attractive way.

Nowadays, computer simulation and 3D visualization techniques play an increasingly important role in scientific activities. The user-friendly Matlab environment, which includes, among others, a high-level programming language and versatile graphics capabilities, helps focus on the applications rather than on programming difficulties. This means that Matlab often becomes the number one choice for scientists, engineers, and students employing computer simulation in their research. Matlab is also a natural choice for visualization planning, mainly due to the efficiency of its data preparation and the availability of OpenGL library functions. There are two main motivations when creating a missile-target engagement rapid visualization software using Matlab: the first is to provide a preliminary, low-cost verification of design conceptions, and the second is to create an environment for 3D visualization of any computer simulation results.

The term *rapid visualization* is adopted from the technique used by graphic artists to create a conceptual drawing in several stages. The purpose of the rapid visualization technique is to take an idea from concept to accurate rendering without having to start over from scratch each time, with the capacity to edit or make alterations, as desired. In the traditional approach, a preferred drawing is created, usually in pencil [5]. Here, the authorial Matlab Battlespace Visualization Class (MBVC) is proposed to plan, design, and compose the preliminary version of the desired computer animation, which can then be used both as a background for the discussion of the final composition and as an individual project for public presentation, providing a visual reference for missile-target engagement simulations.

As a design tool, the MBVC allows us combination of the advanced mathematical models of missile guidance loop elements (implemented in the discrete form as program scripts) with the capabilities offered by open, powerful, and flexible 3D graphical engine. The engine is used for generating both the 3D battlefield environment as well as input signals required for the selected system components (e.g. vision sensors or seekers). In this approach, the 3D visualization software is also a platform capable of integrating program modules developed independently or in parallel by collaborating engineers using unified input-output description schemes. As a visualization tool, the MBVC allows the designers to present their research findings in both an illustrative and informative way, providing a visual reference for the computer simulation. This is especially important when handling complex issues of missile-target engagement [6], optical seekers [7], or some aspects of anti-aircraft system personnel training [8]. Because of its versatility, the software can help to produce attractive presentations for both scientific and military outreach purposes.

The paper is organized as follows. General overview and scene design principles are described in Section 2. To show the features of the proposed solution, sample numerical simulations are performed and their results are given in Section 3. Finally, Section 4 offers some conclusions.

2. SOFTWARE DESCRIPTION

2.1. Software Architecture

The MBVC supports 3D scene composition and visualization, including cameras and lighting sources. By using the functions provided by the OpenGL library, it is possible to texture objects and to use transparency effects. These can be obtained using only basic modules and graphic functions, which ensures the possibility of cooperation with the Matlab environment in various configurations and using an ordinary PC or notebook (a dedicated GPU is required). All the scripts were tested with Matlab R2017a/R2022a under 32/64-bit Microsoft Windows 7, 8.1, and 10 operating systems.

The assumption is that the class will provide the easiest way possible to generate complex 3D scenes. The general idea behind constructing the scene using MBVC is the placing of consecutive objects that act, in effect, like building blocks within the layout. Objects can be created using the *create_object()* method which allows kinematic data, 3D model geometry definitions, and display parameters to be combined into a single easy-to-use data structure. The general procedure scheme for generating a 3D scene using MBVC is presented in Fig. 1.

One of the most important factors conditioning the correctness of the final visualization is the proper preparation of data describing the considered objects (missiles, aerial targets, ground infrastructure elements, etc.) in 3D space. It can be realized using the Cartesian-coordinate matrix **P** and the angular rotation matrix **R** describing the object position and orientation in time:

$$\mathbf{P}=\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_j & y_j & z_j \end{bmatrix} \qquad \mathbf{R}=\begin{bmatrix} \theta_1 & \psi_1 & \varphi_1 \\ \theta_2 & \psi_2 & \varphi_2 \\ \vdots & \vdots & \vdots \\ \theta_j & \psi_j & \varphi_j \end{bmatrix}, \tag{1}$$

where x_i, y_i, z_i are the object gravity centre Cartesian coordinates and θ_i, ψ_i, φ_i are the pitch, yaw, and roll angles in successive instants of time, $i \in \{1, 2, ..., j\}$. For objects whose position does not change over time, the matrices can consist of one row. Data for these matrices must be provided by a user.

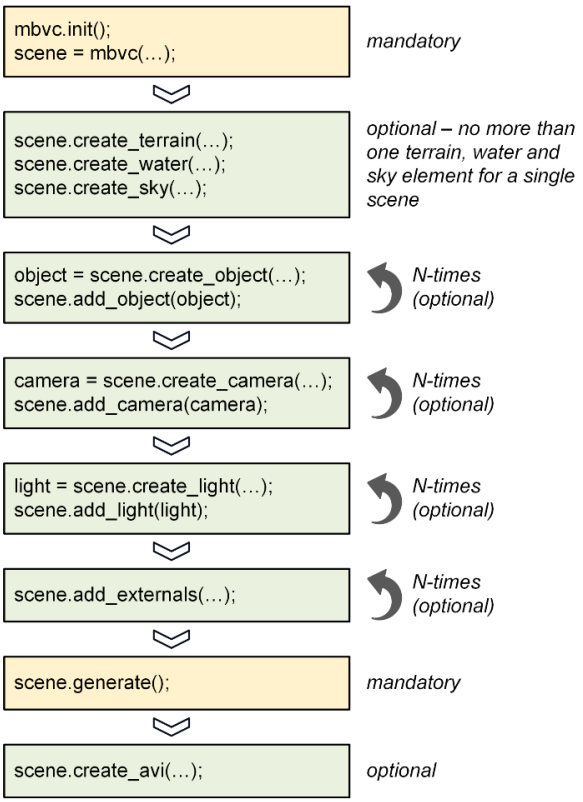

Fig. 1. Procedure scheme for generating a 3D scene using MBVC

The matrix **M** used for object transformations in 3D space is taken as:

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}, \tag{2}$$

in which successive entries are defined as follows:

$$\begin{aligned} m_{11} &= \cos\theta\cos\psi \\ m_{12} &= -\sin\theta\cos\psi \\ m_{13} &= -\sin\psi \\ m_{21} &= \cos\theta\sin\psi\sin\varphi + \sin\theta\cos\varphi \\ m_{22} &= -\sin\theta\sin\psi\sin\varphi + \cos\theta\cos\varphi\,. \\ m_{23} &= -\cos\psi\sin\varphi \\ m_{31} &= \cos\theta\sin\psi\cos\varphi + \sin\theta\sin\varphi \\ m_{32} &= \sin\theta\sin\psi\cos\varphi + \cos\theta\sin\varphi \\ m_{33} &= \cos\psi\cos\varphi \end{aligned} \tag{3}$$

The position and angle values refer to the global inertial coordinate system, which stays in consistent relation to rules used in air defense systems (the azimuth angle equal to zero indicates the north, angle values increase clockwise, etc.).

The class constructor defines the basic parameters of the display window and animation (it can also be called without parameters—then, the default values are assumed). If the use of advanced display options is planned (e.g. texturing or the use of attached 3D models geometries), the MBVC class object creation should be preceded by calling the *init()* method.

Rendering a 3D scene is performed using the *generate()* method. It should be called after adding all planned items to the designed scene. The *generate()* method renders, using the OpenGL library functions, a series of images (frames), the number of which depends on the values of the *display.step* and *timeline* properties specified via the constructor of the MBVC object. These images can then be used as static pictures or as a frameset for creating a movie (the *create_avi()* method is used for this purpose).

2.2. Software Functionalities

The user can attach, to the designed 3D scene item, structures defining: terrain topography and appearance, water level and behavior, sky properties, 3D objects (moving and stationary), as well as light sources and cameras. These items can be added in any order. In the case of moving objects, linear displacement and angular rotation matrices should be specified. It is also possible to dynamically change the positions of light sources and cameras – in this case only the matrices of linear displacements must be defined.

By using the matrix-vector-description scheme, it is possible to modify the parameters of the environment and individual objects during the simulation. This allows, e.g., objects to change their size and color, to appear or disappear, and water levels within a generated scene to raise or lower. All this provides a broad and flexible range of possibilities when designing 3D scenes (Fig. 2).

Creating and adding a camera object to the designed scene results in the automatic switching of the software to from the observer view to the full 3D visualization mode. In turn, creating a multiple-camera system enables us presentation of objects of the scene from different views and efficient switching between them. Cameras can move within the scene, have a defined non-stationary target (look point) and a variable view angle.

Similar rules apply when working with light sources. As in the case of the camera system, the lights can be moved around the scene and characterized by variable color and style (local or infinite). Adding a sky object to the designed scene allows the user to obtain interesting visual effects, e.g., morning or twilight lighting.

Fig. 2. Exemplary 3D scenes created in MBVC

The ability to call external functions and scripts significantly expands the MBVC capabilities. The user-defined functions for the purpose of calling in MBVC do not differ from the standard ones, however, it should be remembered that the variables used in their body must be defined as global (it is not possible to pass them as parameters). The exception is the internal counter of the MBVC class object, which is the only parameter of the function when calling. Attaching external functions is done by the *add_external()* method, which must be called before the start of the scene rendering process.

MBVC supports 3D models stored in MAT file format as the set of filled polygons described by **V** and **F** matrices, where **V** specifies vertex values and **F** defines which vertices to connect. Model geometries can also be imported directly from OBJ files created by popular 3D graphics software, such as Blender or 3ds Max. Please note that the class supports only triangular mesh objects, and before prepared models can be used, appropriate conversions must be made. MBVC includes some predefined models that allow us creation of visualizations without importing external 3D objects—use *help mbvc.create_object()* command for detailed object list.

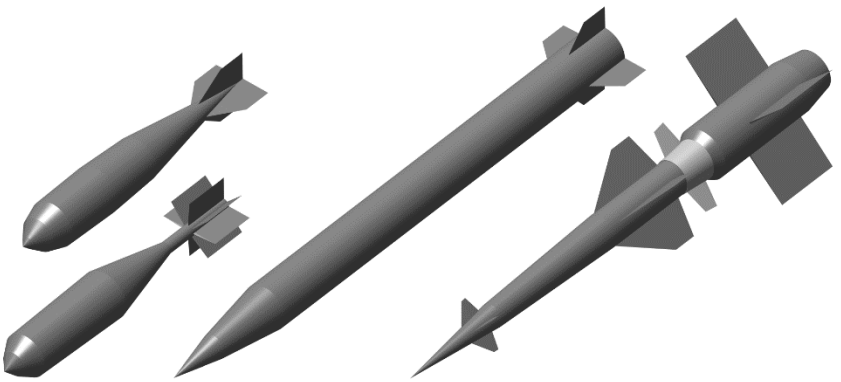

Fig. 3. Axisymmetric model geometries

Moreover, the MBVC includes the *create_model()* method designed to generate parameterized bomb and missile 3D geometries both in MAT and OBJ formats, compatible with the way of describing the data used in the Matlab environment. The function enables us a wide configuration of the designed model parameters, incl. the level of geometry details and the number of control surfaces, which should be sufficient for the needs of most missile-target engagement scenario visualization purposes (Fig. 3).

3. EXAMPLE OF SIMPLE SCENE VISUALIZATION

To demonstrate the main features of the software, example of basic scene visualization will be considered. Skillful combination of the possibilities offered by the MBVC enables us generation of 3D scenes "on the fly", even without conducting sophisticated numerical simulations.

Assume that a simplified non-guided bomb trajectory is described by the following equations:

$$v(t) = v_0 - v_s\left[1 - \exp\left(-\frac{1}{4}t\right)\right], \tag{4}$$

$$s(t) = \begin{cases} x(t) = 0 \\ y(t) = v(t)\cos\gamma \\ z(t) = h_0 + v(t)t\sin\gamma - \dfrac{gt^2}{2} \end{cases}, \tag{5}$$

$$\zeta(t) = \begin{cases} \theta(t) = \tan^{-1}\left[\dfrac{v(t)\sin\gamma - gt}{v(t)\cos\gamma}\right] \\ \psi(t) = 0 \\ \varphi(t) = 0 \end{cases}, \tag{6}$$

where v_0 = 150 m/s is the initial velocity of the system, h_0 = 300 m is the height of dropping the bomb, $\gamma = \pi/6$ rad is the initial angle value, g = 9.81 m/s^2 is the gravitational constant, v_s = 44 m/s is the scaling velocity, and $t \in \langle 0{,}15 \rangle$ s is the time range. The numerical solution of Eqs. (4)-(6) can be easily found using the Matlab code presented below:

```
t = 0:0.01:15; % vector of time instants
v = 150-44*(1-exp(-.25*t)); % vector of velocity instants
P = zeros(length(t),3); % position data matrix, cf. Eq. (1)
P(:,2) = v.*t*cos(pi/6);
P(:,3) = 300+v.*t*sin(pi/6)-(9.81*t.*t)/2;
R = zeros(length(t),3); % rotation data matrix, cf. Eq.(1)
R(:,1) = atan((v.*sin(pi/6)-9.81.*t)./(v.*cos(pi/6)));
```

Exemplary air bomb 3D model geometry is generated using the *create_model()* method as follows:

```
mbvc.create_model('details',80,...
        'diameter',[ 0.00 0.13 0.35 0.48 0.50 0.45 0.00 ],...
        'length',[ 6.09 5.95 5.66 5.27 4.04 2.91 0.00 ],...
        'fins',1,'fin_root_chord',[ 1.5 0.6 0.0 0.0 ],...
        'fin_span',[ 0.00 1.21 1.21 0.00 ],'fin_thickness',0.015,...
        'axis','zxy', 'direction',[ 1 1 1 ],'color',[ .6 .6 .6 ],...
        'output','bomb.mat','show','on');
```

The bomb trajectory is visualized using MBVC methods and the matrices **P** and **R** just obtained:

```
mbvc.init();
world = mbvc('window','fullscreen','toolbar','figure',... % window...
        'timeline',length(t),'step',25,'view',[ 135 35 ],... % ...and scene properties
        'color',[ 1 1 1 ],... % background color
        'xlim',[ -800 800 ],'ylim',[ 0 1600 ],'zlim',[ -200 600 ],... % axis limits...
        'axis','on','xlab','x (m)','ylab','y (m)','zlab','z (m)',... % ...and labels
        'output','local'); % save frameset in local directory
world.create_water('alpha',.6); % add some water to fill the scene
bomb = world.create_object(... % create an object
        'model','bomb',... % use 3D model geometry just created
        'scale',20,... % overscale the bomb to make it visible
        'position',P,'rotation',R,... % position and rotation data matrices
        'face',[ .9 .4 .0 ],'edge',[ .5 .3 .1 ],... % model colors...
        'pathcolor',[ .9 .9 .9 ]); % ...and path behavior
world.add_object(bomb); % add bomb model to the scene
world.generate(); % generate the scene
world.create_avi(); % create AVI file
```

The complete code with comments and all data necessary to perform the visualization is included as an example attached to the MBVC package file. Selected frame of the resulting scene is shown in Fig. 4. To get a fully performed 3D scene, it is only necessary to make a few changes to the parameters of the MBVC object.

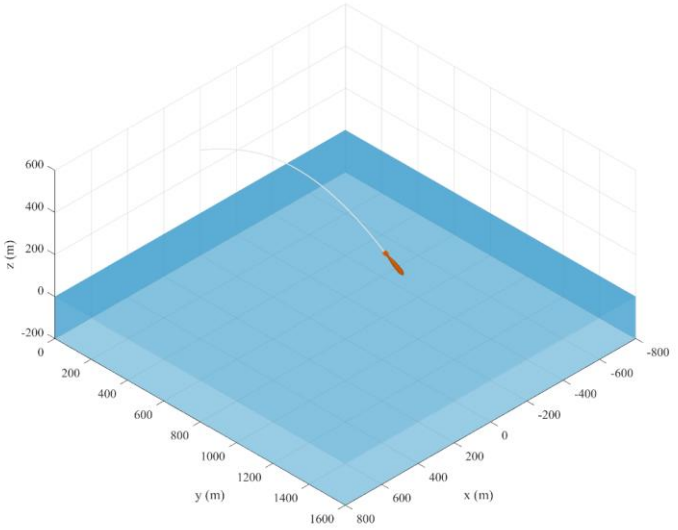

Fig. 4. Sample frame from resulting 3D scene

For example, after adding a camera, creating the sky and turning off the axis visibility:

```
world = mbvc('window','fullscreen','toolbar','figure',...
        'timeline',length(t),'step',25,...
        'xlim',[ -800 800 ],'ylim',[ 0 1600 ],'zlim',[ -200 600 ],...
        'axis','off','output','local'); % axis off
world.create_sky(...% add sky to the scene
        'type','sphere','texture','cloudy',... % use sphere shape and cloudy texture
        'radius',800,'elements',500); % define sphere parameters
world.create_water(...); % same as before
bomb = world.create_object(...); % same as before
world.add_object(…); % same as before
camera = world.create_camera(... % create a camera
        'position',[ 400 1200 400 ],... % set camera position (fixed)
        'target',P,'angle',20); % set camera look point (variable) and view angle
world.add_camera(camera); % add camera to the scene
world.generate(); % generate the scene
```

the result is as shown in Fig. 5.

The design of advanced 3D scenes is carried out in the same way as in the case of the basic example.

The difference is that additional functions are added to the presented code scheme, e.g., for terrain and multiple objects description, scene lighting or camera system (cf. Fig. 1 and user manual included to the software).

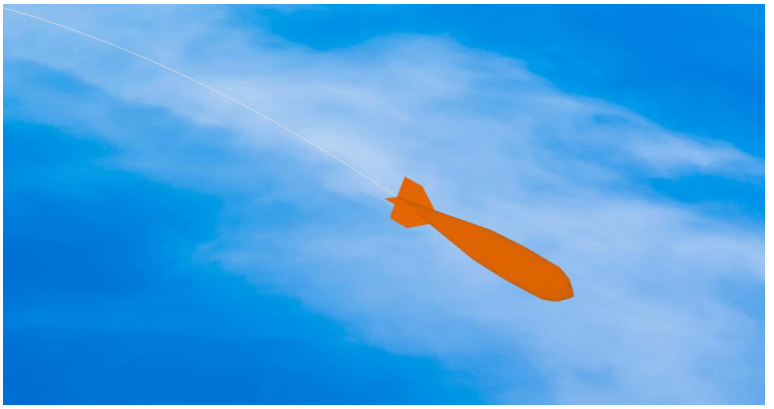

Fig. 5. Exemplary frame from resulting 3D scene

4. CONCLUSIONS

The tool MBVC provides easy-to-use tools for the creation of visual references for computer simulation and result analysis. The class can generate both static images and animations, with user-defined resolutions and parameters. The main area of the MBVC application is related to missiles and aerial targets. However, a general philosophy of use and extensive configuration options enable the MBVC to visualize each object described via position and rotation coordinates in successive instants of time. This makes MBVC an attractive tool to provide 3D visualization in many fields of scientific and engineering activity.

Unlike typical 3D graphics programs, the MBVC provides a direct interface between the simulation environment and the 3D graphics engine. No data conversion is required. This means that the user of the Matlab environment – not necessarily familiar with the secrets of using dedicated 3D visualization software – receives a graphic tool with comparable capabilities, functioning according to known and used paradigms. The tool is noteworthy for its additional functionality, including among others its ability to cooperate with 3D models, create axisymmetric missile geometries “on the fly”, and call external user-defined functions and scripts. Moreover, the software requires only the basic module of the Matlab environment and can be used also with Simulink.

The experience to date shows that this kind of tools is useful in visualization process of the research findings, cf. e.g. [9-12]. The author hopes the MBVC will help present visualizations of a range of research findings in an efficient and attractive way.

One of practical applications of MBVC is the software for computational reconstruction of anti-aircraft artillery firing scenarios. This issue is described in detail in [8].

DISCLAIMER

The software is provided "as is" and the author disclaims all warranties with regard to this software including all implied warranties of merchantability and fitness. In no event shall the author be liable for any special, direct, indirect, or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

The software is still under development (current version: 0.56). Please check regularly for future updates. The MBVC source code is openly available in FigShare Repository at https://doi.org/10.6084/m9.figshare.20593320.v1 under CC-BY-4.0 license.

The described class is freely available for scientific, commercial, and educational use. You can also modify and/or distribute this software for any purpose. However, when using the MBVC, please remember to cite this paper.

FUNDING

The author received no financial support for the research, authorship, and/or publication of this article.

REFERENCES

[1] Zaikang, Qi, and Lin Defu. 2020. *Design of Guidance and Control Systems for Tactical Missiles*. USA, Boca Raton, FL: CRC Press.

[2] Zarchan, Paul. 2012. *Tactical and Strategic Missile Guidance*. USA, Reston, VA: AIAA.

[3] Noga, Janusz, Krzysztof Motyl, Konrad Sienicki, Zbigniew Puzewicz, and Bogdan Zygmunt. 2017. Interception and combat of cruise missiles by Grom MANPADS. In: *Proceedings of SPIE* Vol. 1071502.

[4] Pang, Ce, Gan-lin Shan, Wei-ning Ma, and Gong-guo Xu. 2019. "Sensor radiation interception risk control in target tracking", *Defence Technology* 16 (3) : 695-704.

[5] Baskinger, Mark, and William Bardel. 2013. *Drawing Ideas: A Hand-Drawn Approach for Better Design*. USA, New York, NY: Watson-Guptill.

[6] Nocoń, Łukasz, Marta Grzyb, Piotr Szmidt, Zbigniew Koruba, and Łukasz Nowakowski. 2022. "Control Analysis with Modified LQR Method of Anti-Tank Missile with Vectorization of the Rocket Engine Thrust". *Energies* 15 : 356.

[7] Gapiński, Daniel, and Zbigniew Koruba. 2021. “Control of Optoelectronic Scanning and Tracking Seeker by Means the LQR Modified Method with the Input Signal Estimated Using of the Extended Kalman Filter”. *Energies* 14 (11) : 3109.
[8] Bużantowicz, Witold, and Dariusz Rodzik. 2020. “Computational Reconstruction of an Anti-Aircraft Artillery Firing Scenario”. *Advances in Military Technology* 15 (1) : 125-136.
[9] Ghasemi, A. Kobra, Jafar Ghaisari, and Farzaneh Abdollahi. 2020. “Robust formation control of multiagent systems on the Lie group SE(3)”. *International Journal of Robust and Nonlinear Control* 30 (3) : 966-98.
[10] Rao, Sai Basaveswara, Arnab Chatterjee, and Konstantinos Kanistras. 2022. “System Identification of an Unmanned Aerial Vehicle with Actuated Wingtips”. *Journal of Intelligent & Robotic Systems* 105 : 11.
[11] Heidlauf, Peter, Alexander Collins, Michael Bolender, and Stanley Bak. 2018. Verification Challenges in F-16 Ground Collision Avoidance and Other Automated Maneuvers. In *Proceedings of the 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*. UK, Oxford, July 13, 2018.
[12] Bużantowicz, Witold. 2016. “Matlab Script for 3D Visualization of Missile and Air Target Trajectories”. *International Journal of Computer and Information Technology* 5 (5) : 419-422.

Klasa środowiska Matlab do szybkiej wizualizacji scenariuszy walk powietrznych

Witold BUŻANTOWICZ

Wojskowa Akademia Techniczna
ul. Sylwestra Kaliskiego 2, 00-908 Warszawa

Streszczenie. W artykule przedstawiono autorską klasę Matlab Battlespace Visualization Class (MBVC). Klasa przeznaczona jest do wspomagania projektowania przestrzennych scen wizualizacji i animacji komputerowej, które w dalszej kolejności mogą znaleźć zastosowanie zarówno jako tło do dyskusji nad finalną kompozycją, jak i indywidualny projekt do publicznej prezentacji, zapewniając wizualne odniesienie do wyników symulacji scenariuszy walk powietrznych. MBVC jako narzędzie projektowe umożliwia łączenie zaawansowanych modeli matematycznych elementów obwodu naprowadzania rakiet (zaimplementowanych w postaci dyskretnej w formie skryptów programowych) z możliwościami oferowanymi przez otwarty, wydajny i elastyczny silnik graficzny 3D. Jako narzędzie do wizualizacji MBVC dostarcza projektantom narzędzia niezbędne do przedstawienia wyników badań zarówno w ilustracyjny, jak i informacyjny sposób. Klasa jest bezpłatnie udostępniona do zastosowań naukowych, inżynierskich i edukacyjnych.
Słowa kluczowe: wizualizacja przestrzenna, projektowanie konceptualne, rakieta, cel powietrzny, trajektoria