SoftwareX 22 (2023) 101328

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

viskillz-blender—A Python package to generate assets of Mental Cutting Test exercises using Blender

Róbert Tóth [a,b,*], Bálint Tóth [c], Miklós Hoffmann [a,d], Marianna Zichar [a]

[a] Faculty of Informatics, University of Debrecen, Debrecen, Hungary
[b] Doctoral School of Informatics, University of Debrecen, Debrecen, Hungary
[c] Faculty of Economics, University of Debrecen, Debrecen, Hungary
[d] Faculty of Informatics, Eszterházy Károly University, Eger, Hungary

ARTICLE INFO

Article history:
Received 23 September 2022
Received in revised form 23 January 2023
Accepted 25 January 2023

Keywords:
Blender
Mental Cutting Test
Spatial skills
Assets

ABSTRACT

Several different methods are used to test the spatial abilities or visual skills of people. One of them is the Mental Cutting Test (MCT), the exercises of which offer a 2D projection of a 3D shape and a 2D plane, and testees should determine the shape of their intersection. MCT exercises need various 2D and 3D assets that should be developed before publishing a test. In recent decades, very few exercises have been available to instructors and researchers. In 2019, we published our first solution that could be used to calculate the intersections of MCT scenarios, then render or export their assets using Blender. This paper proposes an extended, open-source package for Blender that generates assets of MCT exercises by permuting predefined shapes, cutting planes, rotation, and scale operators. The additional wrapper script helps users use the package for various purposes, such as developing exercise offering methods, designing exercises, practicing, or organizing exams and measurements.

Code metadata

Current code version	v1
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00301
Permanent link to Reproducible Capsule	
Legal Code License	Apache License 2.0
Code versioning system used	git
Software code languages, tools, and services used	Blender, Python, Miniconda
Compilation requirements, operating environments & dependencies	Python packages `viskillz-common` and `wakepy`
If available Link to developer documentation/manual	https://github.com/viskillz/viskillz-blender/README.md
Support email for questions	toth.robert@inf.unideb.hu

1. Background and motivation

Spatial abilities, such as visuospatial perception, spatial visualization, and orientation, mental folding, mental cutting, and mental rotation play an important role in everyday life as well as in solving complex mathematical problems, and engineering design tasks (see, e.g., [1–7]). The development of spatial skills in the teaching of STEM fields is a challenging task that may require various tools in various pedagogical situations, including planar drawings, real spatial models, and software [8–10]. The primary goal is to map the optimal and effective combination of these tools, including virtual and augmented reality applications, which were proved to be successful in effective development of spatial skills and competences [11,12]. A recent systematic overview of this topic can be found in [13].

There are various, now standard tests to measure these skills. One of these tools is the so-called Mental Cutting Test (MCT), which was originally designed as a part of a college entrance examination test [14]. There are also recent adjustments of this test [15,16], but the classical test is still among the most frequently applied method to assess spatial skills, and the form of tasks are standard in each version.

* Corresponding author at: Faculty of Informatics, University of Debrecen, Debrecen, Hungary.
E-mail addresses: toth.robert@inf.unideb.hu (Róbert Tóth), tothb2000@mailbox.unideb.hu (Bálint Tóth), hoffmann.miklos@inf.unideb.hu (Miklós Hoffmann), zichar.marianna@inf.unideb.hu (Marianna Zichar).

https://doi.org/10.1016/j.softx.2023.101328

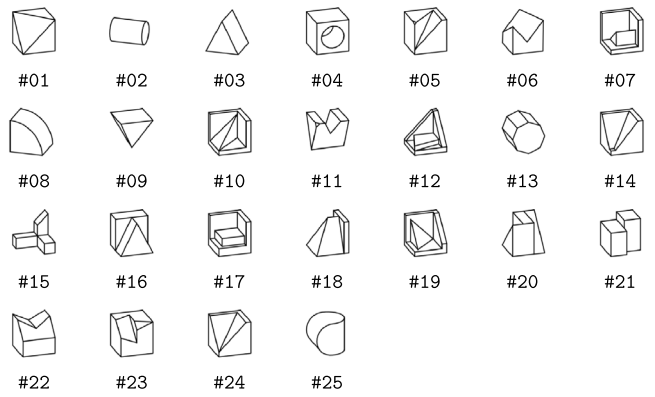

Fig. 1. Reproduced meshes of the well-known sheet of exercises (https://viskillz.inf.unideb.hu/resources/mct-classic.pdf).

An MCT task (also called scenario) contains a 2D projection of a 3D shape and a cutting plane. The exercise seems simple: determine the shape created by intersecting the 3D shape with the plane and select the figure representing the solution out of five candidates. However, scenarios can be constructed with different levels of difficulty. Previous research showed that the difficulty of an exercise strongly depends on the shape and similarity of possible answers [17,18]. If alternatives contain easy-to-recognize differences, a scenario proves to be easier; however, if several alternatives contain very similar features and differ only in minor details, the exercise becomes more complicated.

During the last decades, most researchers and educators used a well-known sheet containing 25 scenarios derived from various shapes and cutting planes, with widely permuted possible answers resulting in a wide range of difficulty levels (see Fig. 1) [19,20]. However, this limited number of available scenarios highly restricts the effective practice and improvement of spatial skills in education. After publishing our script-aided process [21], which aim is to generate resources that allow constructing exercises, we received multiple inquiries to design and construct an open-access database to solve the problem of instructors' and educators' lack of available scenarios.

As the first and most crucial step of the database's development process, package `viskillz-blender` provides an algorithm that enables users to create practically limitless assets using predefined permutation factors. Both 2D and 3D assets of MCT scenarios can be retrieved and serve as input for other methods to find solutions for fundamental multidisciplinary problems that are required to create complete MCT exercises and sheets automatically in the future — to the best of our knowledge, there is no available software for this purpose. The corresponding Blender project contains our intersection planes and manually designed shapes, but users can modify or extend them following the conventions. Based on this environment we can also provide a number of predefined tests, manually selecting 25–25 tasks from the database to assure the presence of all level of difficulty, just as it happens in the original test. The automatic analysis of the difficulty of the assets and the automatic and/or adaptive compilation of the tests will be the task of further research.

2. Methods

To enhance the development of MCT exercises, we have designed an environment that consists of three components:

1. A Blender project, which contains the set of our manually designed meshes and permutation planes.
2. An internal package, which implements the automatic permutation using the Python API of Blender. This package should be located in the folder of Blender, making users able to call its functions directly using Blender's built-in interpreter.
3. A wrapper module, which offers an opportunity to execute goals with the direct use of the Blender's interpreter but invoking the functions of the package as subprocesses.

The core of package `viskillz-blender` is the Blender-based package and its wrapper module. However, a Blender project is needed with the given structure to use the script since it operates on a pre-defined structure, accessing the existing models and creating temporary ones. Thus, we describe the package's features with examples based on our Blender project, which consists of more than 200 manually designed meshes. On the other hand, it can be used as a reference project, allowing users to use its models and add new ones or derive their own projects with the same structure.

3. Blender project

3.1. Original meshes

The process of our research started with the modeling of the 25 classic meshes. Our goal was to minimize the difference between the original images and our models, while some features of the shapes had to be handled carefully to avoid rendering and processing anomalies. On the other hand, the figures of the sheet contain easy-to-recognize errors that can be derived from the sequence of continuous printing and scanning operations that resulted in the current state of the paper sheet. For example, some lines are non-parallel that should be anyway, and relationships between the models and intersections are sometimes broken. Like in our iterative method, our stages are based on the use of Blender's UI, Python API, and its low-level *BMesh* API. For simplicity, 3D shapes are referred to as *meshes*, while 2D shapes as *shapes* or *intersections*.

Each mesh has a maximum length of 2.0 in each dimension, while at least one dimension exactly has a length of 2.0. Thus, our meshes have consistent sizes, which makes the further processing steps easier and deterministic. Origins are also set carefully to the median of each dimension to support rotation transformations. In the case of some meshes, we made a few minor adjustments to avoid containing a large number of edges and vertices in the used cutting planes. Instructors should avoid useless vertices and edges, manually performed subdivision is not required in the case of most scenarios.

However, manually designed meshes can also contain minor errors that can be resulted from floating precision errors. Most models can be designed easily using Blender's *Boolean* modifier, which compute the intersection, the difference and the union of two meshes. During the modeling phase, we detected common miscalculations of this modifier. Thus, we tried to model our meshes without using the *Boolean* modifier, and creating most of the vertices, edges, and faces by starting from simple shapes and applying atomic operations manually to them. However, Blender offers a great set of *Clean Up* methods that are worth applying to our models: *Delete Loose*, *Degenerate Dissolve*, *Make Planar Faces*, and *Split Non-Planar Faces*. There is another important operation in Blender's *Edit Mode*, called *Triangulate Faces*. This operator traverses all the faces and splits them into multiple faces to form a mesh containing only triangles. That step has to be involved in the modeling phase to provide base for better computations.

Each mesh was checked for non-planar faces using the *Split Non-Planar Faces* operator with a limit of 0.5°. Each mesh was modified manually until this operator had no more effect on

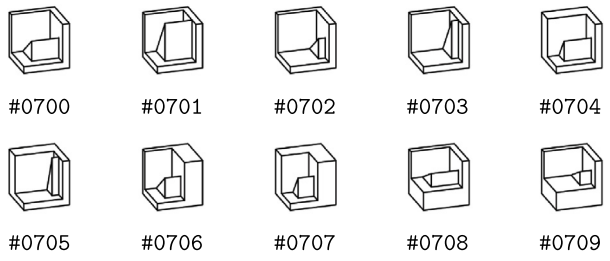

Fig. 2. We changed the main features of mesh #07 to obtain new permutations. As a result, most of the cutting planes will yield different intersections from various meshes having the same orientation.

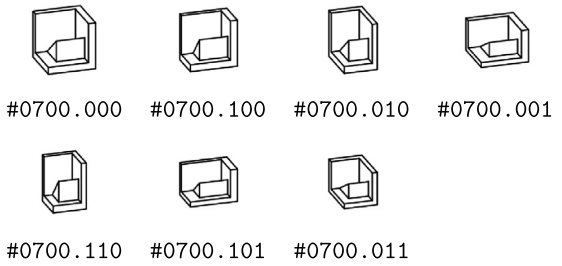

Fig. 3. Automatically created permutations of mesh #07. Additional shapes can be created by applying each scaling vector S on the same, manually permuted mesh.

the mesh, guaranteeing that each face is bent under an acceptable limit. Then each face was triangulated using the operator *Triangulate Faces* with its default parameters. Finally, operators *Degenerate Dissolve* and *Delete Loose* were applied on each mesh, in that order, resulting in clean meshes except for outlier cases caused by the use of the *Boolean* modifier.

3.2. Permute meshes manually

To yield a large set of permutations in the upcoming steps, we derived additional meshes from the original shapes that belong to the well-known MCT exercises. In this step, we changed one or more main features of the original meshes, resulting in very similar alternates that differ only in symmetry or ratios in most cases. On the other hand, additional features are introduced, or an existing feature is deleted to retrieve a permutation. Usually, we created nine alternates of each original shape (see Fig. 2). In the rest of the paper, a mesh will be referred to by its unique ID in format `#GGMM`, where `GG` denotes the ID of the group (derived from Fig. 1) and `MM` denotes the ID of the actual permutation. ID `#GG00` always belongs to the original shape from which the permutations are derived. Thus, `#0100`, …, `#2500` denote the original 25 meshes, while IDs `#0101`, …, `#0109` denote the modified instances that are permuted from mesh `#0100`. A set of permutations will be referred to using the `GG` prefix and called as *shape group* or *group*. The contribution of an instructor ends in this step, except the validation. Thus, all further permutations will be applied in a deterministic, automatized way.

4. The software

4.1. Apply scaling

The features of a mesh – especially the aspect ratio of its edges, thus the aspect ratio of the derived intersections – can be changed by scaling each mesh in one or two global directions, thus we introduce a set of scaling vectors having values 0.7 and 1.0 in different combinations (see Fig. 3). After introducing the seven scaling vectors, a mesh can be referred to with an ID having format `#GGMM.SSS` where the subsequence `GGMM` is inherited from the previous manual permutations, while section `SSS` denotes the ID of the applied scaling vector of the actual shape in which digit `1` denotes scaling factor 0.7, and `0` denotes scaling factor 1.0.

4.2. Apply rotation

Additional permutations can be easily generated from each mesh with the use of rotation transformations. As we are going to intersect each mesh with various planes, the actual orientation of the mesh can change the shape of the intersection. Denote

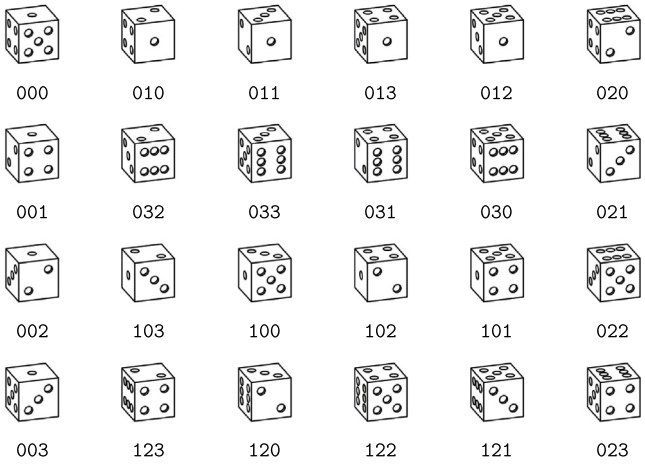

Fig. 4. Different rotations of a dice. The sequence of numbers under each image denotes the unique ID of the rotation in the specified format. With the use of 24 different rotation vectors, each orientation of the cube can be achieved.

the set of rotation vectors as S^{RV} and a single rotation vector as $r = \{X, Y, Z\}$. Each component of any $r \in S^{RV}$ refers to the amount of rotation applied on the corresponding global axis. The amount of rotation is indicated by a value from set $\{0, 1, 2, 3\}$ that is used as the coefficient of 90° when the counter-clockwise rotation along the corresponding axis is performed. For easier notation, we prefer giving literal values in degrees. Thus, each orientation of a shape is referenced with an ID having the format `#GGMM.SSS.RRR` where subsequence `RRR` denotes an element of S^{RV}. Fig. 4 contains orientations of a dice with the ID of the corresponding rotation vector.

4.3. Cutting planes

We already know how the meshes can be permuted, only the set of cutting planes should be chosen. In our iterative algorithm we introduced 19 cutting planes that can be defined with the combination of a point on them and their normal vector in Blender (see Fig. 5). In a real exercise-designing process, we must check whether a cutting plane results in a non-empty, easy-to-understand shape. However, in this paper we focus only on the permutation and our goal is to yield as many different shapes as we can. Thus, all possible combinations of shapes, scaling, rotation and cutting plane should be involved in our computation. To yield more intersections, 12 additional intersections can be derived from the well-known sheet of paper that can result in various intersections (see Table 1).

Table 1
The recently introduced 12 additional planes.

ID	Example	Features	ID	Example	Features
P20		C=(0, 1, 0) N=(0, 1, −1) R=(−45, 0, 0)	P26		C=(0, 0, 1) N=(1, 0, −1) R=(0, 45, 0)
P21		C=(0, −1, 0) N=(0, 1, −1) R=(−45, 0, 0)	P27		C=(0, 0, −1) N=(1, 0, −1) R=(0, 45, 0)
P22		C=(0, 0, 1) N=(−1, 0, −1) R=(0, −45, 0)	P28		C=(0, 1, 0) N=(1, 1, 0) R=(90, 0, 45)
P23		C=(0, 0, −1) N=(−1, 0, −1) R=(0, −45, 0)	P29		C=(0, −1, 0) N=(1, 1, 0) R=(90, 0, 45)
P24		C=(0, 1, 0) N=(0, −1, −1) R=(45, 0, 0)	P30		C=(0, 1, 0) N=(1, −1, 0) R=(0, 90, 45)
P25		C=(0, −1, 0) N=(0, −1, −1) R=(45, 0, 0)	P31		C=(0, −1, 0) N=(1, −1, 0) R=(0, 90, 45)

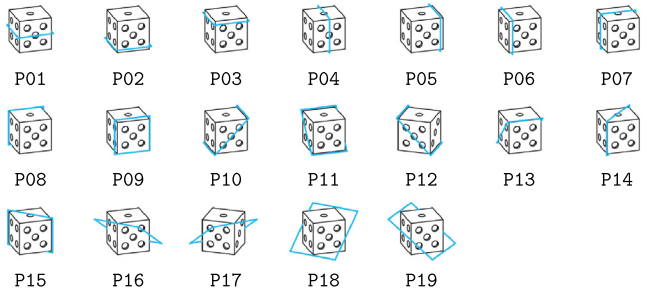

Fig. 5. The previously introduced 19 cutting planes.

4.4. Permute intersections

As a first approach, we can combine each scaling vector with each possible rotation vector, then with each cutting plane. That would yield 13888 combinations for each manually permuted mesh with much redundancy. On the other hand, we can reduce the set of rotation vectors and cutting planes since most of the cutting planes can be omitted from the computations, and substituted with the combination of another cutting plane and rotation vector:

1. P04 and P07 can be substituted with P01;
2. P03, P05, P06, P08 and P09 can be substituted with P02;
3. P11–P15 can be substituted with P10;
4. P17–P19 can be substituted with P16;
5. P21–P32 can be substituted with P20.

Thus, it is sufficient to use only five cutting planes to fetch all the possible permutations. With the help of this information, we can prove that the 64 rotation vectors can be substituted with only 24 ones. Thus, the following set of rotation vectors will be applied on each mesh to yield all the possible permutations:

1. Each rotation vector in which $R_x = 0$;
2. Each rotation vector in which $R_x = 90$ and $R_y \in \{0, 180\}$.

4.5. Compute and export intersections

Given a permuted mesh, a scaling vector, a rotation vector, and a cutting plane, the computation exactly follows the corresponding step of our iterative algorithm. In a nutshell, if a mesh is intersected with a given cutting plane, then the intersection is cleaned and flattened to the XY plane. The shape is translated to the origin and scaled finally. In the iterative algorithm each shape was rendered at this point, resulting SVG and PNG representations. However, we are going to process the coordinates of each edge and its vertices. Thus, instead of rendering, we fetch the local coordinates of each edge and transform them to the global coordinate system. Each rotation vector is used in Blender's XYZ Euler coordinate-system.

To guarantee better precision, each shape is scaled before the calculation of intersections with the vector $\{20, 20, 20\}$, resulting in a shape that has a maximum length of 40 in each dimension. The `bisect()` operator is applied to this state of the mesh. After the post-processing steps, each shape is scaled with vector $\{10, 10, 10\}$ to achieve a maximum length of 20 in each dimension. That guarantees that each coordinate comes from the interval $[-10, 10]$. Finally, the edges of each intersection are dumped into a JSON document to provide a structured dataset for the upcoming Blender-independent steps of our method.

The 25 original meshes have various features, however, the handling of four ones is not obvious if the geometry is described with edges instead of curves. Thus, meshes #02, #04, #06, and #25 are excluded from the generation.

4.6. Goals

With the use of the wrapper module, users are able to specify their workflow in a JSON document, using the following goals:

1. `intersections` – Permutes the shapes of the given groups using all the permutation factors. The output of the stage is a JSON document for each involved group that contains the descriptions of intersections represented by their coordinates.
2. `scenarios-3d` – Permutes the shapes of the given groups using all the permutation factors. The output of the stage is a set of 2D scenarios encoded in GLB assets.
3. `scenarios-2d` – Permutes the shapes of the given groups using all the permutation factors. The output of the stage is a set of SVG scenarios encoded in SVG assets.

A document defines a sequence of goals, and each goal can be parameterized with the required values. The specification of the format and examples can be found in our GitHub repository.

5. Impact

Our environment and its package `viskillz-blender` are designed to support the work of instructors, researchers, and students, who have to deal with Mental Cutting Test exercises. With the use of package `viskillz-blender`, users can generate scenarios including all the 2D and 3D assets needed to construct exercises from multiple scenarios. Moreover, our Blender project serves as a template for a project on which the scripts can be executed; users have an option to use our pre-defined meshes and permutation factors. Moreover, they can derive their projects with the same structure, enabling the script to apply the automatic permutation on their meshes. Our script has already been used in our recent publications, as we have designed a post-processing algorithm that consumes the output of stage `intersections` [22]. Moreover, our *viSkillz Browser* and *viSkillz Quiz* [23] demo applications use assets that were generated with the use of our environment. In parallel, students at our faculty have already chosen thesis topics whose goal is to develop applications based on our dataset created with our package.

MCT exercises have already been manually built using our automatically generated assets. Moreover, the proposed package serves as a base for multiple research directions as well, making researchers able to design and implement their tools to continue the enhancement of workflows related to the field:

- Matching intersections — As the intersections of multiple scenarios can be very similar (or equal), a set of unique intersections should be constructed from which elements can be selected as possible answers for exercises. We are working on a vector-based matching algorithm that compares their edges and vertices using multiple rotation offsets.
- Processing intersections — Understanding the generated scenarios, the set of possible answers should be chosen automatically to offer exercises based on existing scenarios. Various morphological features of each intersection can be retrieved, such as *the perimeter and area*, *the number of vertices and edges*, *the number of regions*, and *the number of lakes and bays*.
- Offering exercises — We are considering data mining and decision-making algorithms to construct exercises. For example, using the K-means family, groups of similar intersections can be created using their features. On the other hand, customizable recommendations are also possible using decision-making systems such as AHP and PROMETHEE [24,25]. We have already used the mentioned algorithms to recommend programming exercises [26], but researchers can consider many other solutions from the field of data mining, decision making, and machine learning [27].
- Encoding assets — The significant number of permutations results in more than 1,000,000 scenarios, requiring more than 6 GB of storage space just for the GLB assets of our dataset. Encoding schemes should be created to support the lossless storage and efficient serving of assets [28].

Finally, the output allows researchers to organize measurements, evaluate the efficiency of MCT exercises, and use different perspectives, 3D, AR, and VR solutions. For example, we have already performed a pilot measurement in our faculty based on exercises created from the automatically generated assets [23]. This quiz contains a sample test that consists of 10 exercises. We are also going to publish additional sample tests with form and level of difficulty analogous to the original test, that can be used to evaluate the method before implementing automatic exercise offerings.

6. Conclusions

In this paper, we introduced our *viSkillz Blender* project, which aim is to support users in creating assets for Mental Cutting Test exercises. The result of the development is package `viskillz-blender`, which can apply automatic permutation on manually designed meshes in Blender using a set of intersection planes, scaling, and rotation transforms. Then, three different assets can be retrieved automatically: the 3D shape and the cutting plane as a classic 2D asset or in its 3D version, and their computed intersection as the possible answer. Moreover, we also publish the reference Blender project, which can impact other research directions such as offering exercises and encoding assets.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Robert Toth reports a relationship with Ministry for Innovation and Technology that includes: funding grants. Miklos Hofmann reports a relationship with Ministry for Innovation and Technology that includes: funding grants.

Data availability

No data was used for the research described in the article.

Acknowledgments

Supported by the ÚNKP-22-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funders.

References

[1] Bohlmann N, Benölken R. Complex tasks: Potentials and pitfalls. Mathematics 2020;8(10). http://dx.doi.org/10.3390/math8101780.
[2] Bishop AJ. Spatial abilities and mathematics education—A review. Educ Stud Math 1980;11(3):257–69. http://dx.doi.org/10.1007/BF00697739.
[3] Tosto MG, Hanscombe KB, Haworth CMA, Davis OSP, Petrill SA, Dale PS, et al. Why do spatial abilities predict mathematical performance? Dev Sci 2014;17(3):462–70. http://dx.doi.org/10.1111/desc.12138.
[4] Cole M, Wilhelm J, Vaught BM-M, Fish C, Fish H. The relationship between spatial ability and the conservation of matter in middle school. Educ Sci 2021;11(1). http://dx.doi.org/10.3390/educsci11010004.
[5] Zimmermann W, Cunningham S, Mathematical Association of America. Committee on Computers in Mathematics Education. Visualization in teaching and learning mathematics: A project. MAA Notes and reports series, Mathematical Association of America; 1991.
[6] Raju G, Sorby S, Reid C. Investigating the relationship between spatial skills and engineering design. In: ICERI2022 proceedings. 15th annual international conference of education, research and innovation, 2022, p. 421–6. http://dx.doi.org/10.21125/iceri.2022.0144.
[7] Sorby SA. Eucational research in developing 3-D spatial skills for engineering students. Int J Sci Educ 2009;31(3):459–80. http://dx.doi.org/10.1080/09500690802595839.
[8] Presmeg N. Visualization and learning in mathematics education. In: Lerman S, editor. Encyclopedia of mathematics education. Cham: Springer International Publishing; 2020, p. 900–4. http://dx.doi.org/10.1007/978-3-030-15789-0_161.
[9] Presmeg N. Spatial abilities research as a foundation for visualization in teaching and learning mathematics. In: Clarkson P, Presmeg N, editors. Critical issues in mathematics education: Major contributions of Alan Bishop. Boston, MA: Springer US; 2008, p. 83–95. http://dx.doi.org/10.1007/978-0-387-09673-5_6.

[10] Gerber A, editor. Spatial abilities. A workbook for students of architecture. Basel: Birkhäuser Verlag GmbH; 2020.
[11] Alpiste Penalba F, Torner Ribé J, Brigos Hermida M. Exploring virtual reality to improve engineering students' spatial abilities. Pilot study. In: EDULEARN19 proceedings. 11th international conference on education and new learning technologies, IATED; 2019, p. 6275–84. http://dx.doi.org/10.21125/edulearn.2019.1503.
[12] Huerta O, Unver E, Arslan R, Kus A, Allen J. An approach to improve technical drawing using VR and AR tools. Comput-Aided Des Appl 2019;17(4):836–49. http://dx.doi.org/10.14733/cadaps.2020.836-849.
[13] Papakostas C, Troussas C, Krouska A, Sgouropoulou C. Exploration of augmented reality in spatial abilities training: A systematic literature review for the last decade. Inform Educ 2021;20(1):107–30. http://dx.doi.org/10.15388/infedu.2021.06.
[14] CEEB. CEEB special aptitude test in spatial relations (MCT). Ddveloped by the college entrance examination board. 1939.
[15] Quaiser-Pohl C. The mental cutting test "Schnitte" and the picture rotation test-two new measures to assess spatial ability. Int J Test 2003;3(3):219–31. http://dx.doi.org/10.1207/s15327574ijt0303_2.
[16] Cohen CA, Hegarty M. Inferring cross sections of 3D objects: A new spatial thinking test. Learn Individ Differ 2012;22(6):868–74. http://dx.doi.org/10.1016/j.lindif.2012.05.007.
[17] Németh B, Sörös C, Hoffmann M. Typical mistakes in Mental Cutting Test and their consequences in gender differences. Teach Math Comput Sci 2007;5(2):385–92. http://dx.doi.org/10.5485/TMCS.2007.0169.
[18] Nagy-Kondor R, Esmailnia S. Polyhedrons vs. curved surfaces with mental cutting: impact of spatial ability. Acta Polytech Hung 2021;18(6):71–83. http://dx.doi.org/10.12700/APH.18.6.2021.6.4.
[19] Gorska R, Sorby S, Leopold C. Gender differences in visualization skills - An international perspective. Eng Des Graph J 1998;62(3):10.
[20] Nemeth B, Hoffmann M. Gender differences in spatial visualization among engineering students. Ann Math Inform 2006;33:169–74.
[21] Tóth R. Script-aided generation of Mental Cutting Test exercises using Blender. Ann Math Inform 2021;54:147–61. http://dx.doi.org/10.33039/ami.2021.03.011.
[22] Tóth R, Tóth B, Zichar M, Fazekas A, Hoffmann M. Detecting and correcting errors in Mental Cutting Test intersections computed with Blender. In: Cheng L-Y, editor. ICGG 2022 - Proceedings of the 20th international conference on geometry and graphics. Cham: Springer International Publishing; 2023, p. 904–16. http://dx.doi.org/10.1007/978-3-031-13588-0_79.
[23] Tóth R, Tóth B, Zichar M, Fazekas A, Hoffmann M. Educational applications to support the teaching and learning of mental cutting test exercises. In: Cheng L-Y, editor. ICGG 2022 - Proceedings of the 20th international conference on geometry and graphics. Cham: Springer International Publishing; 2023, p. 928–38. http://dx.doi.org/10.1007/978-3-031-13588-0_81.
[24] Saaty T. The analytic hierarchy process: Planning, priority setting, resource allocation. Advanced book program, New york: McGraw-Hill International Book Company; 1980.
[25] Brans J, Vincke P. A preference ranking organization method. Manage Sci 1985;31(6):647–56.
[26] Adamkó A, Kádek T, Kollár L, Kósa M, Tóth R. Cluster and discover services in the Smart Campus platform for online programming contests. In: 2015 6th IEEE international conference on cognitive infocommunications. 2015, p. 385–9. http://dx.doi.org/10.1109/CogInfoCom.2015.7390624.
[27] Tan P-N, Steinbach M, Karpatne A, Kumar V. Introduction to data mining. 2nd ed. Pearson; 2018.
[28] Tóth R, Hoffmann M, Zichar M. Lossless encoding of mental cutting test scenarios for efficient development of spatial skills. Educ Sci 2023;13(2). http://dx.doi.org/10.3390/educsci13020101.