Mathematical and Software Engineering, vol. 5, no. 2 (2019), 23-33.
Varεpsilon Ltd, http://varepsilon.com

New Functionalities of a Virtual Computer Model Design and Construction

Stoyan Stoyanov[1] and Stanimir Zhelezov[2]

[1] Konstantin Preslavsky University of Shumen, Shumen, Bulgaria,
email: 404i90@gmail.com
[2]Konstantin Preslavsky University of Shumen, Shumen, Bulgaria,
email: s.zhelezov@shu.bg

Abstract

The purpose of this paper is to construct a virtual model of a computer system for student training, based on analysis of different platforms, creating virtual reality and its related 3D graphics. A classification of three-dimensional models was made. Software platforms for three-dimensional modelling and creation of virtual scenes have been analysed and selected. A virtual model has been developed.

Keywords: virtual reality; 3D–modelling; virtual scenes

1 Introduction

The purpose of the current paper is based on analysis of different platforms, for creating virtual reality and its related 3D graphics, and creating a virtual model for student training in the subjects Computer assembling and Computer architecture. For that objective the following tasks should be solved:

- Considering the principles of computer architecture, the basic components of a computer system and the assembly of a computer system, and common assembly errors;
- Analysing the principles of 3D computer graphics and virtual modelling;
- Choosing the right toolkit based on the analysis of the most popular 3D modelling and virtual reality tools;
- Developing an application for training needs.

2 Assembling a Computer System

Computer architecture is a specification that describes how a set of software and hardware technology standards interact to form a computer system or platform. In short, computer architecture refers to how a computer system is designed and what technologies it is compatible with. Computer architecture was proposed by the mathematician John von Neumann in 1945. He describes the design of an electronic computer with its processor, which includes arithmetic logic device, control unit, registers, data and instruction memory, input-output interface and external memory [1, 15].

In order to assemble a computer system, the components of such a system must be determined and whether the components given are consistent. The main computer hardware components that are found in every modern computer are: motherboard, central processing unit (CPU), random access memory (RAM), power supply, video card, hard disk drive (HDD), optical device (for example BD / DVD / CD device), monitor, keyboard, and mouse.

When assembling a computer system, the appropriate components should be carefully selected in terms of their compatibility. Improper selection of components leads to risks of damage and "burns", and as is well known the cost of all the above components is quite high. Therefore, it can cause serious financial losses. The training of students [2, 3] in the subject Computer Assembling, which is one of the major ones in Computer Science direction, is related to the use of a serious financial resource to purchase the necessary set of components. In addition, during training, incorrect assembly of computer systems may result in additional costs. Even with proper assembly, the so-called Von Neumann bottleneck can appear, which slows down the PC performance and is difficult to locate since the components are compatible with the motherboard but there is a difference between the specifications of the two components.

This is the reason why the topic of this paper is focused on virtualization of the whole process of hardware assembly of the CS. For this purpose, it is necessary to properly select tools and software for both 3D modelling and virtualization of 3D models. Additional functionalities, such as compatibility of hardware components, checking for the presence of a positioned object, etc., should also be implemented, which also depends on the choice of the programming environment. This selection is discussed in the next paragraph.

3 Modelling 3D objects and selecting virtualization software

Each 3D image created by a computer requires three main components:

1. Description of a three-dimensional scene;
2. One or more light sources;
3. Camera description or scene view.

Scene description usually consists of one or more models or three-dimensional structures. Usually, one thinks of a model as a standalone piece, such as a pencil or a tree, and the scene as an assembly of these parts in a complete three-dimensional environment. This attitude influences the most common procedure for building a three-dimensional scene: many models are built and then assembled. Each model has two descriptions [4]: mathematical representation of the structure of the form and view of the shape when illuminated.

The structural description is basically geometry. It tells us where the object is in space and where it is not. Most computer programs use the built-in computing system in today's computers. This hardware has very high but limited precision. Some modelling methods are very close to this approach; they create points in the space or cut it very finely and label it blank or full. Other methods try to allow the model to be described more abstractly, such as that a glass is a cylinder with a bottom.

The other part of each model is the description of the surface. This is to describe the physics of the interaction of the surface of the model with the light. It turns out that this physics is guided by several descriptive terms that are intuitively relevant to humans, such as colour, brilliance and transparency [4]. Since models often simulate real-world shapes, it must be determined how detailed the model should be [5, 6]. Three-dimensional modelling is the process of creating a three-dimensional representation of any surface or object by manipulating polygons, edges, and vertices in a simulated three-dimensional space [7].

Nowadays, modelling results can be seen everywhere. Movies, animations, and video games that are filled with fantastic and imaginative creatures and structures. Three-dimensional modelling can be accomplished manually with specialized three-dimensional production software that allows the artist to create and manipulate polygonal surfaces or by scanning real objects into a set of data points that can be used to digitally represent a model that is capable of being fully animated, making it an essential process for animation and special effects [8]. The essence of the model is the grid that is best described as a collection of points in space.

These points are mapped into a three-dimensional mesh and joined together as polygonal shapes, usually triangles or squares. Each point or vertex has its own position in the mesh, and by combining these points in shapes, the surface of an object is created [4, 7]. Models are often exported to other software for use in games or movies. However, some 3D modelling programs allow the creation of 2D images using a process called 3D rendering [5,9]. This technique is fantastic for creating hyper-realistic scenes using sophisticated lighting algorithms [9]. The model can also be physically created using the so-called 3D printers. In addition to the surfaces of the model, texture mapping can be added to make the object more realistic.

When constructing three-dimensional models, one of the following techniques is being used: box/subdivision modelling, edge/contour modelling, NURBS/spline modelling, digital sculpting, image-based modelling, 3D scanning, and procedural modelling.

Box/Subdivision modelling
Box modelling is a polygonal modelling technique in which the artist starts with a geometric primitive (cube, sphere, cylinder, etc.) and then refines its shape until the desired look is achieved. Artists using this modelling technique often work in stages, starting with a low-grade grid, refining the shape and then subdividing the grid to smooth the hard edges and add detail [7,9]. The subdivision and refinement process is repeated until the grid begins to contain enough polygonal details to convey the intended concept properly. Box modelling is probably the most common form of polygonal modelling and is often used in conjunction with edge modelling techniques [12].
Edge/Contour modelling
Edge modelling is another polygonal technique, although fundamentally different from box modelling. In edge modelling, instead of starting with primitive shape and refinement, the model is essentially constructed piece by piece, placing polygonal faces along contours and then filling all the gaps between them [10]. This may sound unnecessarily complicated, but some grids are difficult to be completed just by modelling boxes, such as the human face. Proper face modelling requires very strict edge control of the mesh, which shapes it, and the precision provided by contour modelling can be invaluable [11]. Instead of trying to shape a well-shaped cavity from a polygonal cube (which can be confusing and counter-intuitive), it is much easier to build the outline of the eye and then model the rest. After modelling the main parts of the human face (eyes, lips, forehead, nose, jaw), the other parts tend to fall into place almost automatically [10,12].
NURBS/Spline Modelling
NURBS is a modelling technique mostly used in automotive and industrial modelling. Unlike polygonal geometry, the NURBS grid has no faces, edges, or vertices. Instead, NURBS models consist of smoothly interpreted surfaces created by a lofting grid between two or more Bezier curves (also known as splines) [9]. NURBS curves are created with a tool that works very much like a pen tool in MS paint or Adobe Illustrator. The curve is drawn in 3D space and edited by moving a series of handles called CVs (control vertices). To model the surface of NURBS, the artist places curves on visible contours and the software automatically interpolates the space between them. Alternatively, a NURBS surface can be created by rotating a profile curve around

the central axis [13]. This is a common (and very fast) technique for modelling objects that are radial in nature - wine glasses, vases, plates and more.

Digital Sculpting

The tech industry likes to talk about some of the breakthroughs they call disruptive technologies. Technological innovations that change the way we think about achieving a particular task. The automobile has changed the way we drive. The Internet has changed the way information and communication are accessed. Digital sculpting is a technology that allows you to create intuitively three-dimensional models in a way very similar to sculpting a "digital clay" sculpture.

In digital sculptures, grids are organically created using a (Wacom) tablet device to shape the model almost exactly like a sculptor who would use brushes on a real piece of clay [7]. Digital sculpting has taken the modelling of characters and creatures to a new level, making the process faster, more efficient, and allowing artists to work with high-resolution grids containing millions of polygons. Sculpture grids are known for previously unimaginable levels of surface detail and natural (even spontaneous) aesthetics.

Image-Based Modelling

The image-based modelling is the process by which the transformable three-dimensional objects are algorithmically extracted from a set of static two-dimensional images [12]. Image-based modelling is often used in situations where time or budget constraints do not allow the creation of a fully realized 3D asset manually. Perhaps the most famous example of image modelling is in *The Matrix*, where the team has neither the time nor the resources to model full 3D kits. They shot action scenes with 360-degree cameras and then used an interpretive algorithm to allow "virtual" motion of 3D cameras through traditional real-life scenes.

3D Scanning

3D scanning is a method of digitizing real-world objects when incredibly high levels of photorealism are required. A real-world object (or even an actor) is scanned, analysed, and data (usually an x, y, z point cloud) is used to generate an accurate polygonal or NURBS grid [13]. Scanning is often used when requiring a digital representation of a real actor, as in *The Curious Case of Benjamin Button*, where the main character (Brad Pitt) ages in the opposite direction.

Procedural Modelling

The word *procedural* in computer graphics refers to everything generated algorithmically, rather than created manually by the artist. In procedural modelling, scenes or objects are created based on user-defined rules or parameters [12].

In the popular Vue, Bryce and Terragen environmental modelling packages, entire landscapes can be generated by setting and changing environmental parameters such as leaf density and altitude range, or by selecting landscapes such as the desert, the Alps, coasts, etc.

Procedural modelling is often used for organic structures such as trees and greenery, where there are almost endless variations and complexities that would take a lot of time (or impossible) for the artist to capture by hand [14]. The SpeedTree application uses a recursive / fractal-based algorithm to generate unique trees and shrubs that can be modified by adjusting trunk height, branch density, angle, curl, and dozens, if not hundreds of other options. CityEngine uses similar techniques to generate procedural urban landscapes.

When selecting the software to build the 3D model needed for virtualization of the computer configuration, the most common 3D modelling software systems were analysed - Maya, Cinema 4D, LightWave, Modo, Houdini, Blender and 3ds Max [10].

Each of the systems under consideration has a number of advantages such as a powerful animation package, support for wide format import and export, a rich list of modelling tools, and more. Of course, they also have many drawbacks - poor particle system, insufficient intuition, and delay in high-detailed models, poorly organized interface and more. Some of these systems have a large set of tools and prefabricated objects, but they have disadvantages such as high cost, huge hardware requirements, steep learning curve, and more. On the other hand, free software is poor from built-in tools and objects, they are quite unstable, slow down their work on more detailed models, etc.

From the analyses performed of the 3D modelling software systems, Cinema 4D stands out the most suitable [11]. Some of the main advantages over the others are:

- Clear interface: Customizable floating and configurable dashboards allow space to be dedicated to the work, not the interface.
- Cinema 4D is known for its accessibility. Easy to use by artists and designers, as well as by users without their prior knowledge of the field.
- Mograph is an incredible toolkit in Cinema 4D that speeds up the process of creating animations and graphics for motion.
- Newly developed Irradiance Cache.
- Updated Bevel tool.
- Intel Embree in a physical renderer.
- Extremely stable.
- Cinema 4D is fully loaded with a rich library of predefined objects

Cinema 4D is intuitive and handy for both professional designers and beginners. It comes with an extremely large range of features and tools that make object modelling time significantly shorter. It renders comprehensive documentation that gives a full description provided by a number of examples, as well as video and text lessons and live training sessions. It also provides a friendly interface that makes it easy to build 3D models. Last but not least, the program provides an opportunity for easy remodelling of objects in order to facilitate the construction of alternative three-dimensional objects, which is one of the goals of this paper in terms of future development [12].
For the implementation of the 3D model of the computer system (Fig. 1), models of the basic elements of the system are built - motherboard, power supply, hard disk and box (Fig. 2).

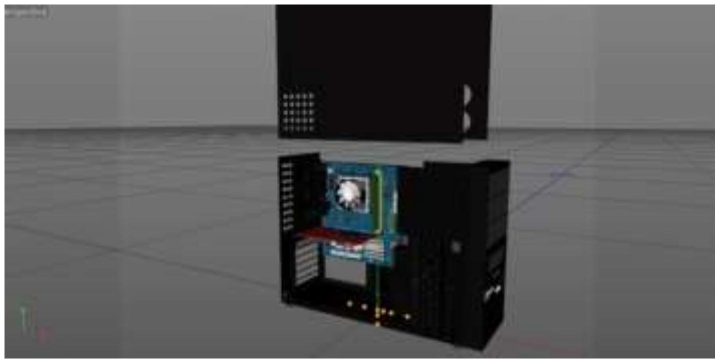

Figure 1 Three-dimensional model of a PC

For the creation of the main objects additional sub-objects (their heirs) are being used, which are constituent parts of the object, and texture of the object. The motherboard is a collection of different types of slots (Fig. 2) and the components located on these slots - video card, processor, RAM, coolers, etc. (Fig. 3). And also various small details - capacitors, etc. (Fig. 2)

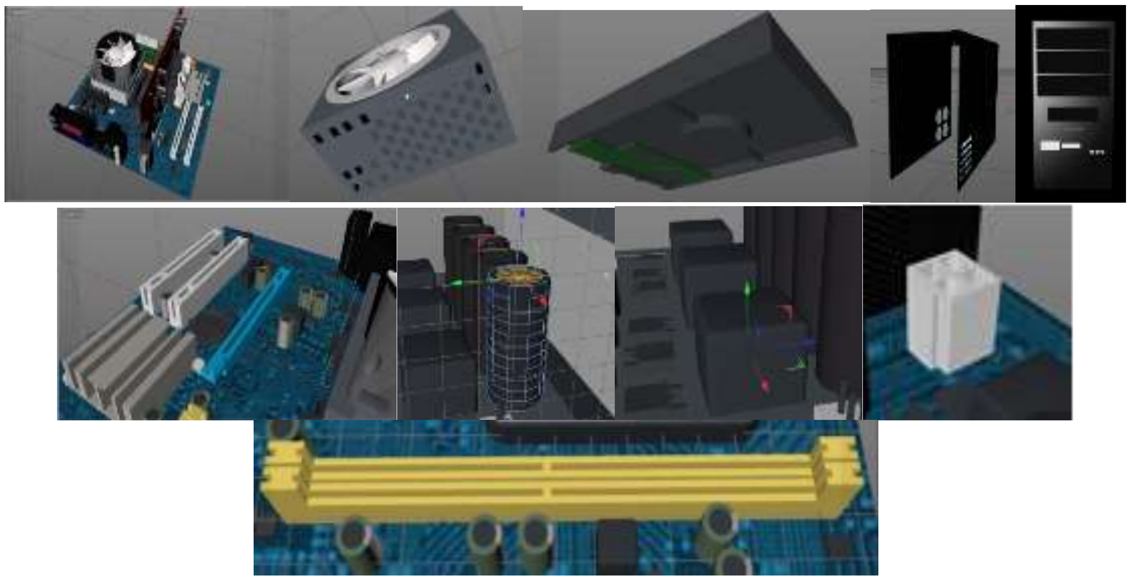
Figure 2 Main elements of the motherboard model

Each of these components can also be considered as a separate object. This allows the computer system configuration to be modified when modelling an additional set of components. This can increase the number of configurations available to the trainees.

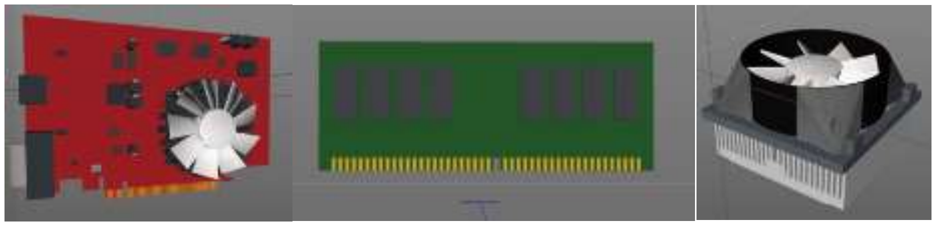
Figure 3 Components of the computer system

In order to improve the application, by creating a number of component compatibility and slot-checking systems and others to create a sense of even greater realism than before, additional computer components were created, from several types of processors, RAM, video cards and more that will be included in the compatibility and employment systems of the motherboard.
To improve the player's experience and facilitate their work, a workshop was set up (Fig. 4) to house the additional components and the computer system.

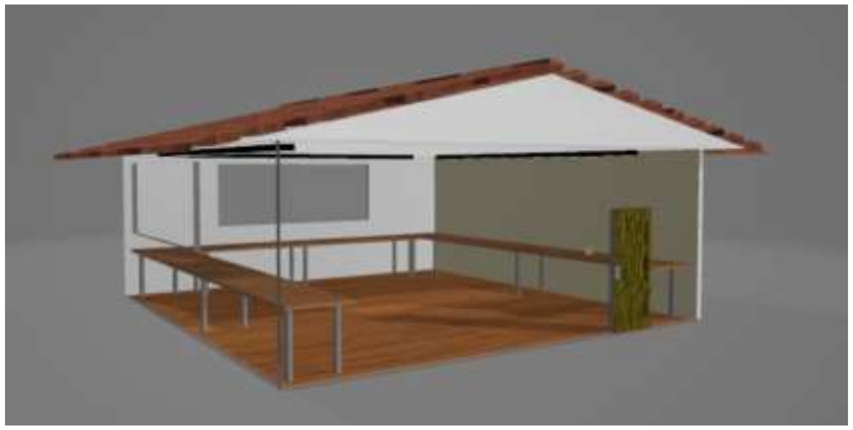
Figure 4 Workshop in finished form and ready for export

4 Virtualization application implementation

After the successful modelling of the object, comes its import into the virtual environment. As with the choice of modelling environment, a comparative analysis of different virtualization software platforms has been made. The advantages and disadvantages of the most common products - Unity and Unreal Engine have been compared. To realize the virtual scene, the Unity software platform was selected [13]. Its advantages as an excellent balance of ease of use and power, built-in physically based rendering and high quality graphical effects, active development, regularly fixed bugs and regularly released new features, etc., determine its choice for the virtual scene. The ultimate goal of the application is to assemble and disassemble a computer configuration in a virtual environment. For this purpose, the model is exported from Cinema4D to Unity and is imported into the program using the Unity's built-in tools while maintaining the detail of the three-dimensional model. After successful export/ import, the individual components of the model are "created" as moving objects so that the player can lift and leave them in different positions. One major component named Mesh Collider (Fig. 5) is added to all major model objects (cover, motherboard, hard disk drive, power supply, processor, processor cooler, RAM). Mesh collision is used on complex objects with inaccurate shape, and it automatically takes the shape of the object.

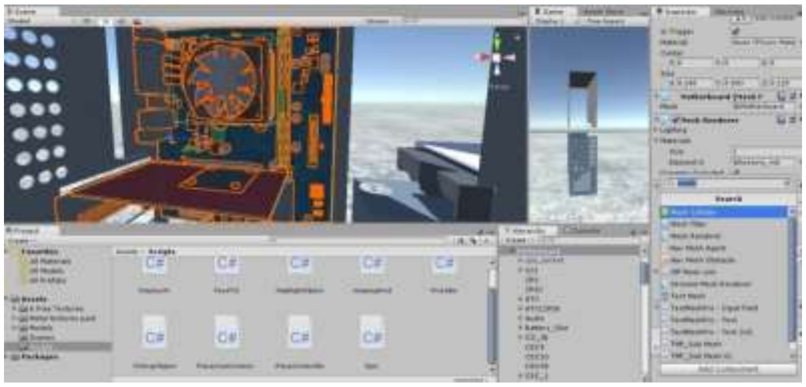

Figure 5 Mesh collider on the components of the computer, that is, they can now interact with the virtual environment

The model and its sub-objects need a mesh collision in order to be lifted, left, etc., otherwise the collision would be static without the collision. A C# script is created to allow individual model objects to be raised, left, or rotated. By adding a function to the already created script, information is added to each object that has been selected (Fig. 6).

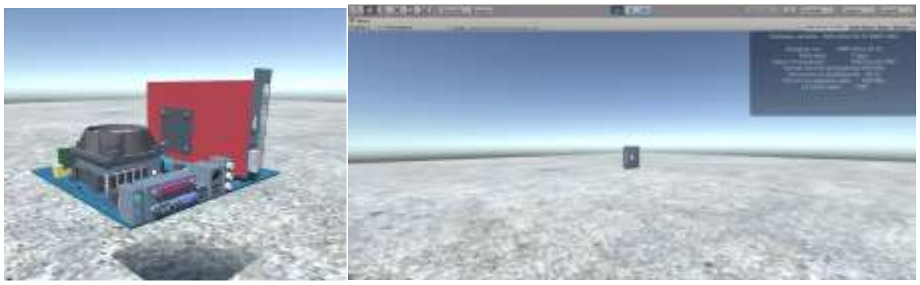

Figure 6 Motherboard and processor after removing and adding information

After the components that were added to the computer system were created, they were exported to the virtual scene. The ultimate goal of the application is to add compliance/ compatibility and employment options to the already ready for assembly/ disassembly, browsing and site information, as well as to improve the current ones.

First step:
The exported object, which will be a workshop (Fig. 7), is sized to fit the size of the player and the computer components and placed on an already constructed plane in the virtual scene. The same is done for the new components, after which they are placed on the tables of the workshop.
The Box Collider component is added to the walls and tables of the workshop to enable them to be interacting objects. Otherwise the components would fall through the table and the player would go through the walls. Box Collider can be seen in (Fig. 7) with the thin green lines along the edges of the table.
Second step:
Adding Highlight Object and Hover Text functions (Fig. 8). Until now, the application lacked these features and the player only had a sight in the center of the screen and could hardly tell when they were pointing to one component, which in turn made the "picking up" of components difficult. The addition of these features aims to improve the player's conditions and experience in the simulation.

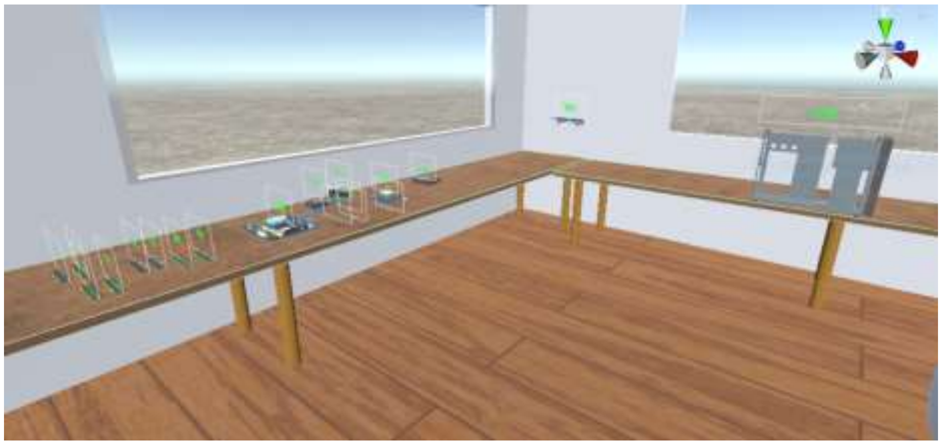
Figure 7 Workshop located in the plane with stacked components

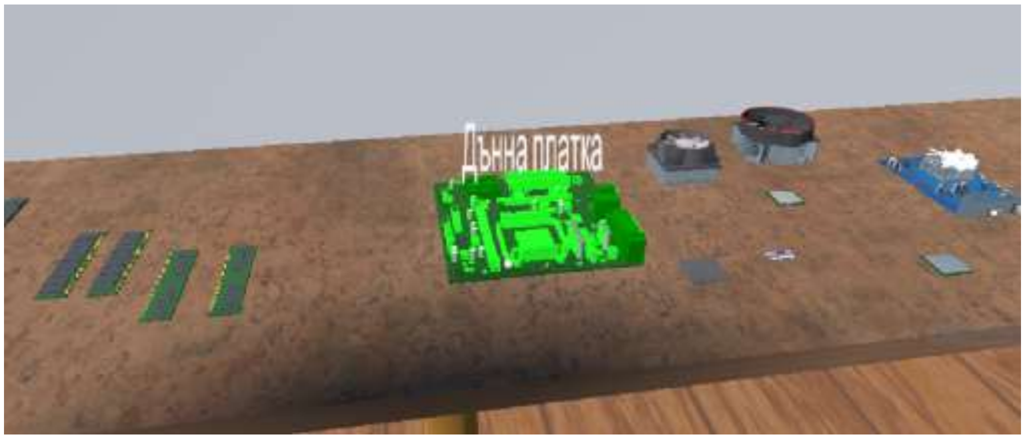

Figure 8 The Highlight Object features colour the object in green, indicating that the object is specified, and Hover Text provides information about the specified object

The next step was to improve the placement of components on the motherboard or the computer case. Because only the global coordinates were stored for the component so far, this meant that if the parent object was moved anywhere randomly on the stage, then the object with the predefined coordinates would go where the parent object was located before moving it. To prevent this from happening, the local coordinates are already being saved. The component placement button was also removed. The distance between the given component and the slot it meets is checked instead and the component adheres to the given slot when the condition is met. Along with the adherence of the objects, the function for checking the compatibility of the components was added (Figs. 9 - 10). As noted in (Fig. 11), there is a check for consistency between two objects using an additional CheckComp script that uses a public string variable predefined by the Unity inspector panel for each object to which slot they are compatible.

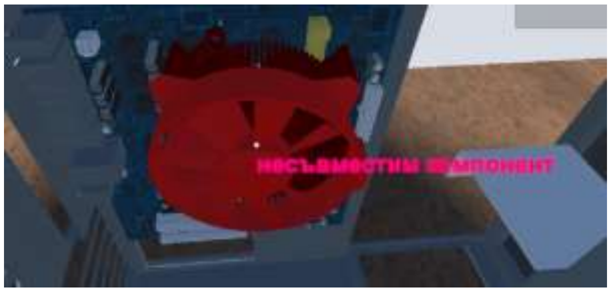

Figure 9 When the object approaches the slot corresponding to the object, if it is incompatible, the object turns red and announces its incompatibility

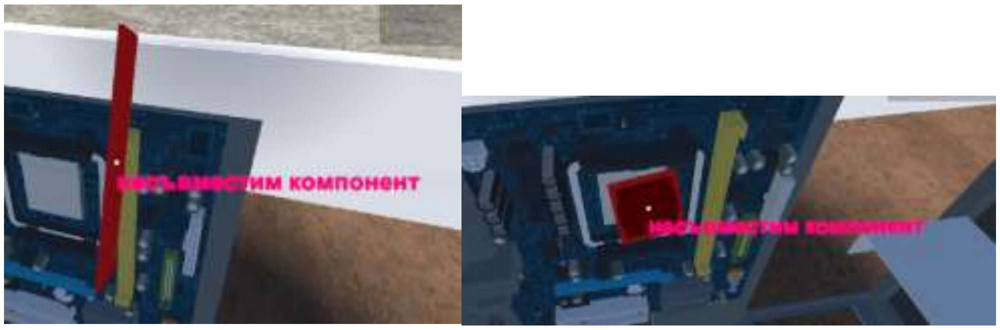

Figure 10 DDR3-type RAM incompatible with the motherboard that requires DDR2 and Intel i7 processor incompatible with the AM2 processor slot

The addition of the following functionality employment of the slot (Figs. 3.2.7 - 3.2.9) was implemented similarly to the incompatibility function by checking whether another object (component) is no longer present in the slot for which the given object, “carried” by the player, is intended.

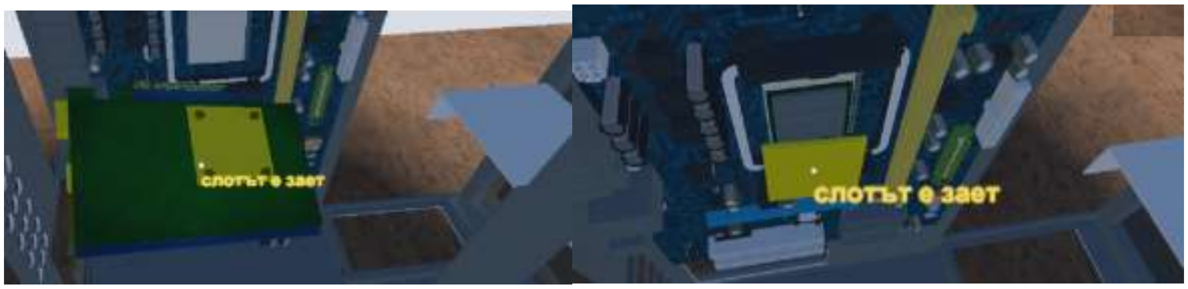

Figure 11 When the object approaches the slot corresponding to the object, if the object is already inserted, the object carried by the player turns yellow and announces the presence of a component in the slot

5 Conclusion

The study of hardware-oriented courses, such as Computer Architecture and Assembly, requires a large amount of technical resources. Their presentation to students as hardware components leads to purely practical difficulties in terms of creating premises and repositories of this type of visual material. To avoid these inconveniences, the capabilities of virtual reality systems can be used.

The developed virtual scene and the associated three-dimensional model can serve as a good basis for the subsequent construction of complete virtual systems for assembling and disassembling computers. With the help of these systems, students could, regardless of their location, acquire the necessary knowledge and skills for the hardware of various types of computer systems and their parameters, characteristics and compatibility. The proposed three-dimensional model can be easily modified in order to expand the base of visualization components and thus to represent the development of computer technology over the years with visual means.

Acknowledgements

The research presented in this paper is partially funded by the project of Shumen University: RD-08-96/01.02.2019.

References

[1] Stanev, S. (2013) Computer Architectures, Module 1. Computer Organization and Architecture. Module 2. Fundamentals of assembler programming for microcomputer simulators. ISBN 978- 954-577-761-5, Shumen, 2013, p. 320.

[2] Stoyanov, B., Kordov, K. (2014) Open Source Software Alternatives in Higher Education Following Computer Science Curricula 2013. Research Journal of Applied Sciences, Engineering and Technology, Vol. 8, No. 9, pp. 1160-1163, ISSN: 2040-7459.

[3] Malchev, D., Kordov, K. (2014). WEB-based distance learning training system. MATTECH 2014 SCIENTIFIC PAPERS, VOLUME 1, pp. 149-154, Bishop Konstantin Preslavsky University Publishing House, ISSN: 1314-3921.

[4] Haresh Kh. (2008) Applications of CAD Software: What is Solid Modelling? www.brighthubengineering.com.

[5] Wisslar, V. (2013) Illuminated pixels: the why, what, and how of digital lighting. Cengage Learning.

[6] Gahan, A. (2012) 3D Automotive Modelling: An Insider's Guide to 3D Car Modelling and Design for Games and Film. Focal Press.

[7] Ratner, P. (2012) 3-D human modelling and animation, Van Nostrand Reinhold, ISBN-13: 978-0442025083.

[8] Wissler, V. (2013) Illuminated Pixels: The Why, What, and How of Digital Lighting, 2013.

[9] Hristova, R. (2017) An Example of Using Computer Animation in Astronomy Lessons (In Secondary School), Naukoviy chasopis Natsionalynogo pedagogichnogo universitetu imeni M. P. Dragomanova, Seriya 3. Fizika i matematika u vishtiy i seredniy shkoli, Vipusk 18, Kiiv, 131-136, ISSN 2410-7816.

[10] Savelonas, M. A., Pratikakis, I., and Sfikas, K. (2015) An overview of partial 3D object retrieval methodologies, Multimedia Tools and Applications 74.24: 11783-11808.

[11] Earnshaw, R. A. (2014) Virtual Reality Systems. Academic press.

[12] Stoyanov, S., Zhelezov, S. (2018) Designing and Constructing a Virtual Computer System Model, MATTECH 2018, Volume 1, pp. 205-212.

[13] Roedavan, R. (2014) Unity Tutorial Game Engine, Bandung: Informatika.

[14] Ebert, D., Musgrave, K., Peachey, P., Perlin, K., and Worley, S. (1994) Texturing and Modelling: A Procedural Approach.

[15] Velcheva, K., Velcheva, V., and Stanev S. (2008) Virtual model of MAIN FRAME COMPUTER ES 1020B from Computer Museum of Mathematics and Informatics Faculty in Shumen University. In Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing (CompSysTech '08), Boris Rachev and Anger Smrikarov (Eds.). ACM, New York, NY, USA, Article 85 . DOI=http://dx.doi.org/10.1145/1500879.1500973.