Interactive Assembly Simulation in an Immersive Virtual Environment

Cătălin Moldovan
Technical University of Cluj-Napoca
Str. G. Bariţiu 28, 400027, Cluj-Napoca, România
catalin.moldovan97@gmail.com

Adrian Sabou
Technical University of Cluj-Napoca
Str. G. Bariţiu 28, 400027, Cluj-Napoca, România
adrian.sabou@cs.utcluj.ro

DOI: 10.37789/rochi.2020.1.1.22

ABSTRACT
This paper describes the development of a Virtual Reality (VR) simulation application for educational purposes in assembly production. People are able to retain more information by simulating real experiences. Nowadays, modern technology can be used to achieve better results in a virtual reality learning environment, being available to everyone. It is predicted to be more common and low-priced in the future. Interactive user gestures and immersive VR technologies are used to develop remote solutions for engineering students by matching the product components in the proper order and location. This research provides instructional assembly methods and natural experience in interacting with elements, both in a dynamic space or in a sitting position.

Author Keywords
Virtual Reality; Leap Motion; Assembly Simulation; OpenGL; Educational Application;

ACM Classification Keywords
H.5.m. Information interfaces and presentation: Human-Computer Interaction; Interaction Techniques; Gestures;

General Terms
Virtual Reality; Computer graphics; Algorithms; Input devices; Head-mounted display;

INTRODUCTION

Modern graphic techniques have changed the way people perceive the virtual world, developing a new approach of understanding problem solving. Many devices and computer software allow us to convert human interaction into tridimensional (3D) data, in order to replicate the human hand model on screen. So much time is being waste to learn a new user interface of an application for a novice computer user. People can adapt faster in an immersive experience, than using a desktop application. The experience is achieved from the first movements and gestures performed by the user, without a tutorial or guided texts. The virtual contact with the environment is based on user's natural interaction intuition.

Virtual reality users are growing from day to day, from 200.000 users in 2014, to over 171.000.000 users in 2018 [1]. It has become widely spread in the game industry and now is expanding on manufacturing and medical industry. The first Oculus Rift prototype was released in 2012 by Palmer Luckey and the game engine designer of Doom franchise, John Carmack. This start was noticed by many big companies and quickly made their way into business industry, becoming available to the public.

Nowadays many educational facilities, like mechatronics, machine building, aviation or construction technical college are missing essential equipment to train students. In many cases, students know the theoretical part and even the main process, but this data is forgotten in time due to lack of real demonstration or simulation. The objectives of this paper is to increase learning performance by using modern technology to simulate real-life assembly. The graphics environment is designed on a long-term human retention, based on visual and interactive processes. Simple structures can create an attractive and innovative space scene, so that users can better perceive the objects' depths and distances, based on 3D placement and shadows [2].

The main keys are accuracy and user comfort. Having these in mind, environment accommodation during the virtual simulation will no longer create confusion or motion sickness, while wearing a headset. The graphic scene rendered on the screen is designed to send the mind into a more immersive 3D experience, having full control on the virtual space.

After a virtual experience session, students are more trained to apply what they learned in real situation. Within a virtual assembly session, people might try to explore wrong alternatives, deviating from the base assembly process, which might lead to bad consequences, like breakage, nonfunctional devices, short circuit or even accidents. We cannot experience wrong possibilities in real life, but in a simulation, everything is possible, leading the user to identify the correct assembly process and to understand the incompatibility of certain parts.

The proposed solution makes use of the head movements and natural hand interactions of user's actions, in order to simulate a virtual model assembly, using tracking devices like sensors and cameras. This paper elaborates on models orientation, which tries to simulate object manipulation by human in real world and presents a fresh new graphic engine, as the project's foundation. The VR application was built

using OpenGL library with the Oculus Rift Development Kit 2 (DK2) headset and LeapMotion controller for hand motion tracking. The main focus is aimed on simple and complex techniques of interaction, in order to reach the behavior of a real life object manipulation.

The rest of the paper is structured as follows. In the next section, we present similar ideas and projects designed by students and researchers. In the following four sections we describe the major graphics engine infrastructure components, the interaction algorithms developed for object manipulation, parent-child relationship between two objects and an overview of the 3D model creation. In the next two sections we discuss about different guided methods of model assembly and the performance of level completion. In the last section we describe the experience gained from implementing the overall assembly project, we present our conclusions and future project improvements.

RELATED WORKS

Zhao et al. [3] present a VR simulation game for manufacturing education by interacting with LEGO pieces, using wireless controller in hand. The overall goals of the project is to provide engineering students with a set of scenarios to practice their skills at craft production. They describe the development of the immersive experience by using a custom fitted headset with Tobii eye-tracking technology. The environment is based on assembly station for the users to go through and accomplish a set of requirements. The user has to choose the components in order to start crafting the production process.

Pujol-Tost and Phil [4] analyze the influence of computational VR interactivity in the learning process, based on response speed, range of things that can be changed and naturality of communication. They analyze them all by showing how they involve different learning and interaction strategies. The source of motivation in the learning process has proven to be higher, the more interactive and immersive the experience is. The main key point is the equality of conditions when user interacts with the content, simulating similar real experience. As formal educational environments have demonstrated a positive attitude towards interactive devices, they continue to evolve and be more accessible to people all over the world.

Zimmons and Panter [5] proceed an experiment of college-age participants on how visual elements like lighting, surface detail and task performance influence the sense of presence of participants in a virtual environment. Based on some graphics conditions, the experiment uses a head mounted display and a joystick, with a trigger function to grab objects from scene. The study suggested that rendering quality environments is not significantly affecting the perception of depth or user's precision. A major difference of spatial orientation was determined not to be equal between man and women.

Pop and Sabou [6] use the LeapMotion controller to interact with virtual scene, using Unity Game Engine. They present an approach to dynamic data visualization and manipulation through a server-side application, based on hand gestures and head movement and orientation, tracked from phone's gyroscopic information.

Galais et al. [7] evaluated gestural interaction using LeapMotion and a traditional interaction device, using gamepad controllers. The comparative study is based on the cognitive load and performance of object manipulation, performed by 11 experienced users and 8 novice users. The results indicate a higher execution time and users' errors during gestural interaction with the LeapMotion device rather than using a controller. The main limitations are intermittent hand tracking and the difficulty in interacting and reaching the object as no haptic feedback is provided.

Boud et al. [8] conducted a series of experiments to compare assembly completion times after participants study an engineering drawing or an assembly plan, using VR and Augmented Reality (AR) as training media. In order to achieve simple goals of interacting with objects, like reaching an object, grasping or placing objects, which require different levels of haptic and visual guidance. A VR manufacturing environment allows users to manipulate objects without the use of the real objects and also to be trained for an assembly operation during a product's design cycle, before an actual physical prototype has been manufactured. The participants suggested that immersive VR was more intuitive as they were able to manipulate 3D objects in a 3D space. AR can therefore facilitate fast learning for simple assembly tasks, as it allows the user to have tactile feedback through the manipulation of the real objects.

Baggett and Ehrenfeucht [9] present how to design instructions that show and describe a step by step procedure using a hierarchical structure. The structure of an object can be represented by a labelled tree, as each node has a value, which presents the object's name. The tree shows the model breakdown into subassemblies and subsubassemblies, the procedure description, which tells the actions performed and the goal to build the complete model, which can be divided in subgoals. The paper tests the performance in assembly from memory, as the object is correctly built by the user. The best performance is achieved when combining a top-down approach with a sequential execution of actions. It is also demonstrated that the presentation of instructions via a video can improve performance of assembly operations. Humans have a remarkable ability to store visual information over short periods of time. Simply seeing the assemblies being built was sufficient for experienced participants to be able to develop assembly plans.

GRAPHICS ENGINE

Creating a lightweight graphic engine for this project, focused on render algorithms and interaction methods, might

be useful for the freedom of using minimum computer resources. The free real-time 3D creation platforms Unity and Unreal Engine 4 offer a user interface, many options and properties for use to design and conceptualise the virtual world. Figure 1 displays the engine specifications of different free engines. Code files, models and materials are efficient organized for the user and the real time application scene makes designing easier and faster. Visual scripting technique lets user create scene content events without coding skills. Both engines have many plugins and scene creation tools available on their asset store. The complexity as well as the numerous integrated features contribute to the final application size, providing additional specifications which are not always needed.

Our application is based on free C++ libraries for graphic software developing, based on OpenGL Shading Language (GLSL). The efficiency in using a fresh new engine, is based on extensions, quality optimization and memory allocation. As follows, there are also disadvantages of building a custom engine as speed processing, data partitioning, threads execution model or the number of features. The application is designed as a flexible tool based on virtual interaction structure organized in a software architecture, having the possibility to study system response to external hardware, resource management and 3D transformation concepts.

Godot Engine is a free and open-source game engine which at first sight, it would be the best choice of developing a small application, aimed on 2D and simple 3D games. The issue might be more of scaling, which might affect the performance, but overall it is not at the level of support, features and functionality compared to other engines. It has its own programming language GDScript, but similar to our solution, the application's configuration has to be made manually by the user [10].

The overall engine solution comes with visual effects for lightning, shadow mapping, environment mapping, reflective materials properties creation, text and video rendering, Table 1. Sounds and animation elements were used for focusing user attention on the action location. The main limitations identified are mainly focused on the scene realism, low on extensions and complex application structure.

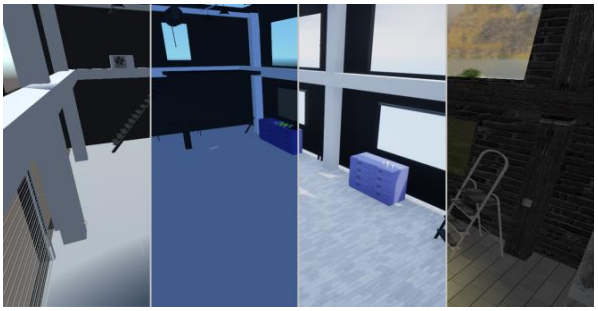

Figure 1. Graphic Engines scene comparison, from left to right: Unity, Godot, Unreal Engine, our engine solution

Engine / Service	Proposed Engine	Unity	UE4	Godot 3.0
Programming language	C	C#	C++	GDScript C/C++ C#
Framework	OpenGL	Direct3D OpenGL Vulkan	Direct3D	OpenGL
Dimensions	3D	2D, 3D	2D, 3D	2D, 3D
Storage Space	130 MB	4 - 7 GB	10-15 GB	500 MB
VR support	Yes	Yes	Yes	Yes

Table 1. Engines specifications

The immersive components

The overall VR session is based on the communication between human and hardware components. The immersive environment is achieved by synchronizing the hand interaction, head position and orientation with the virtual world, having at least 60 frames per seconds displayed on the headset's screen. Communication between user and system is done through input devices, by sending the human movement and interaction information to the computer and output devices, which receive the processed data back to the user. The human head is traced by the camera-based system, which uses filters to capture infrared light trackers on the back of the Oculus headset case. LeapMotion sensors and the monochromatic camera allow the user to interact within the virtual scene. The software is processing each human hand bone, tracked in the device's range and store them as data, which can be accessed by an API for each available frame processed. The information is used to trigger scene events, recognize hand gestures and render the skeleton of the human hand model into the scene.

The virtual hand system is built of geometric shapes, which recreate the hand bones anatomy. For each finger presented in Figure 2, we associate four cylindrical bodies, that are used for representing bones length and four sphere bodies, which connect them together, resulting the skeleton shape of the hand. Tracking algorithms interpret the data and deduce the positions of the undetectable hand elements from the Leap sensors, to ensure a continuous presence of the virtual hands on screen, as long as possible.

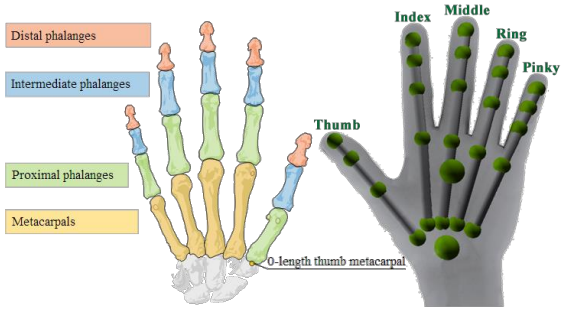

Figure 2. Hand anatomy [11] and virtual model used in app [12]

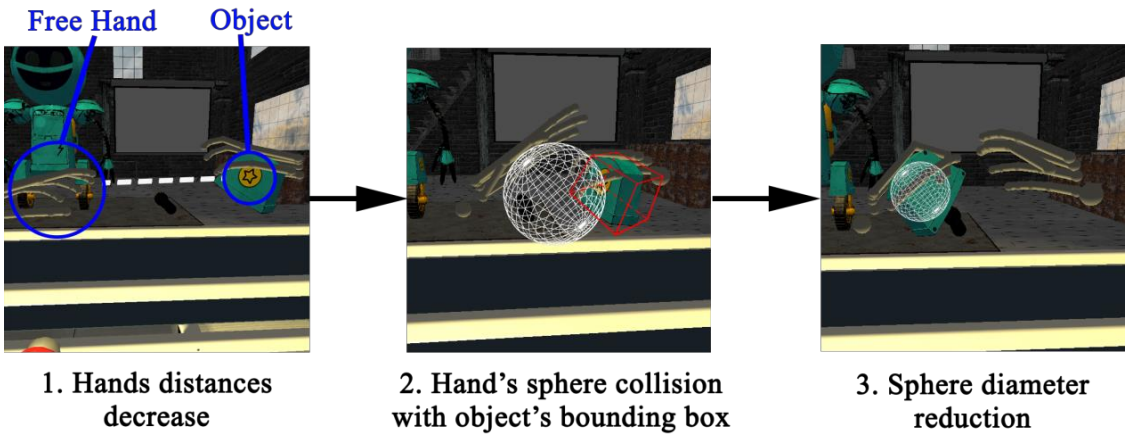

Figure 3. The steps of changing objects from one hand to another hand

The main code path of the VR application is executing a loop in which the Oculus camera sensors request the headset position, then creates the scene texture for each eye. The stereoscopic sensation is operated automatic by the Oculus SDK. The final rendered scene is post processed for each frame by Oculus Compositor, in order to apply distortion and then it is displayed onto the Rift's screen [13].

INTERACTION

Each device has their own coordinate system, which has to be synchronized, in order to be correctly displayed on the screen. Leap Motion tracking software processes the human hand on its visual angle, then Oculus library render the scene and place the virtual hand. model on its coordinate system. In order to use the LeapMotion device attached on the Rift headset, some operations are needed for placing the hand system in front of the virtual camera. As the Leap Motion company does not provide a mirrored hand system technique, all bones and joints have to be manually oriented, by flipping the hand information, received from Leap API, on the local Z axis [14]. Rotating the system at 90° on user's local X axis, will result in rendering the human hand motion in the intended place, similar to real interaction. These operations are also needed to be applied on objects, when interacting with them, in order to maintain the same coordinate system as the hand. When using LeapMotion device on a surface, there is no need of this correction anymore.

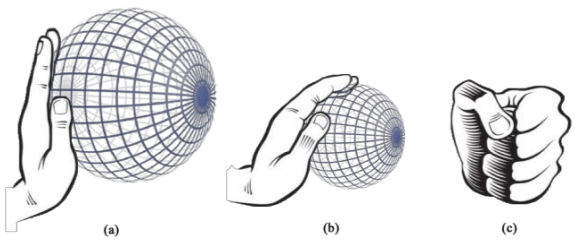

Figure 4. Hand elements of grabbing objects from scene, sphere diameter 180mm (a), ideal grab event 90mm (b), 0mm (c)

Leap SDK offers controller positions, rotations, normals and other data as 3D coordinates vectors. Hand rotation is computed based on hand direction, as the distance from the middle of the palm towards the fingers, and palm normal, as a vector pointing downward of the palm [15]. Based on palm's information, user can grab objects from the scene just by clenching their fist. For that, we create a virtual sphere which covers the length of the fingers, Figure 4. The sphere is placed roughly as if the hand was holding a ball. As fingers are closer to the palm, the sphere radius is reduced and when is used near an object, the grabbing event is triggered. The object is linked to the center of the sphere so that it gives the impression of holding it. In order to interact with a part of the assembly model, a free user hand has to be near the object. The sphere diameter is tested so as not to exceed a constant value, which matches the 45° hand angle. Touching the collision mesh of the object causes the link between hand and object. Both hands can be used simultaneously to interact with the scene and also to move objects from one to another hand, see Figure 3.

Complex gestures are available to be used by experienced users, to rotate model parts directly in the hand, without placing and grabbing them again from conveyor belt. This technique uses two hands, one is holding the object and the other one is performing gestures, in order to rotate the object based on the movement direction, Figure 5. To access free object rotation mode, user will perform a pinch gesture, with the other three fingers raised up, so that it is not interpreted like a grabbing gesture.

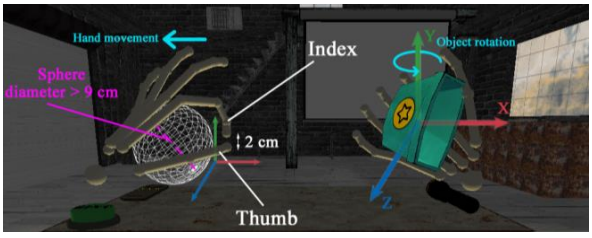

Figure 5. Complex gesture local coordinate system

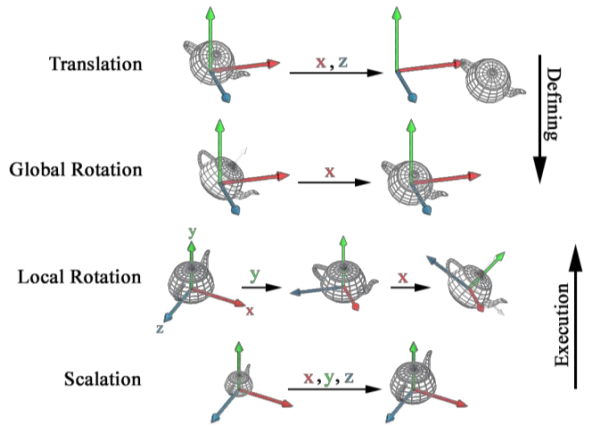

Figure 6. Object transformation order in application code

Overall, the object interaction is composed of 3 levels of rotation. When grabbing an object from scene, their coordinate system is attached to the corresponding virtual hand system, so that each tilt performed with the hand in any direction of the axis will be followed by the object. As the object can be grabbed from the conveyor belt in any direction user want, the rotation has to be made on global axis, after the local rotation computation, see Figure 6. All the operations made on the object are computed starting from the identity matrix, Equations 1 and 2, where m_{00}, m_{10}, m_{20} represent the X axis coordinate, m_{01}, m_{11}, m_{21} represent the Y axis coordinate, m_{02}, m_{12}, m_{22} represent the Z axis coordinate.

$$M_{identity} = \begin{pmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & \mathbf{1} \end{pmatrix} \quad (1)$$

$$M_{projection} = \begin{pmatrix} \boldsymbol{m_{00}} & \boldsymbol{m_{01}} & \boldsymbol{m_{02}} & m_{03} \\ \boldsymbol{m_{10}} & \boldsymbol{m_{11}} & \boldsymbol{m_{12}} & m_{13} \\ \boldsymbol{m_{20}} & \boldsymbol{m_{21}} & \boldsymbol{m_{22}} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix} \quad (2)$$

Beside their original importing process into scene, objects can be discarded in any posture when they touch the belt. The orientation is retained and recalculated throughout the entire running application, based on the final projection matrix of the object, Equations 3, 4 and 5, where α, β, γ are the roll, yaw and pitch angle rotations [16].

$$\alpha = arctg\left(\frac{m_{10}}{m_{00}}\right) * \frac{180}{\pi} \quad (3)$$

$$\beta = arctg\left(\frac{-m_{20}}{\sqrt{{m_{21}}^2 + {m_{22}}^2}}\right) * \frac{180}{\pi} \quad (4)$$

$$\gamma = arctg\left(\frac{m_{21}}{m_{22}}\right) * \frac{180}{\pi} \quad (5)$$

The final layer of rotation is acquired by hand gestures, which is added to the local object rotation. The movement performed by the hand will be mapped on the next rotation axis of the object. An example is presented in Figure 5, when moving the left hand on X axis, will result in rotating the object around Y axis. This way, the object manipulation will rotate on the respective axis of the user performing the hand movement, making intended rotation behavior. All layers put together will result in a predictive system response to the human hands motion.

Beside their original importing structure into scene, objects are oriented based on the current state of rotation on the conveyor belt, LeapMotion sensor's horizontal angle of view and simple and complex hand interactions.

Algorithm 1. Hand-object matrix transformation

```
GETOBJECTLOCATIONANDORIENTATION()
.IDENTITYMATRIX()
.TRANSLATION(palmSphereCenter)
.ROTATION(90 * toRadians, AXIS(1,0,0))
.ROTATION(180 * toRadians, AXIS(0,0,1))
.ROTATION(pinchMovement.z, AXIS(1,0,0))
.ROTATION(pinchMovement.x, AXIS(0,1,0))
.ROTATION(pinchMovement.y, AXIS(0,0,1))
.ROTATION(hand.direction().pitch, AXIS(1,0,0))
.ROTATION(hand.direction().yaw, AXIS(0,1,0))
.ROTATION(hand.palmNormal().roll, AXIS(0,0,1))
.ROTATION(obj.rotation().x, AXIS(1,0,0))
.ROTATION(obj.rotation().y, AXIS(0,1,0))
.ROTATION(obj.rotation().z, AXIS(0,0,1))
.SCALING(obj.scale())
.RENDERMODEL()
```

CHILD OBJECTS

Interacting with a virtual world is not always friendly, because your real body is not transferred in the new environment. The presence of an avatar, which illustrate the human body, may give the impression of trust and also provides distance approximation of the virtual world. We attach a 3D model to the camera, so that looking down to the feet, the user will see parts of the model, see Figure 7 (a). Each user movement performed in real life will be followed in moving the virtual body model with the camera's position.

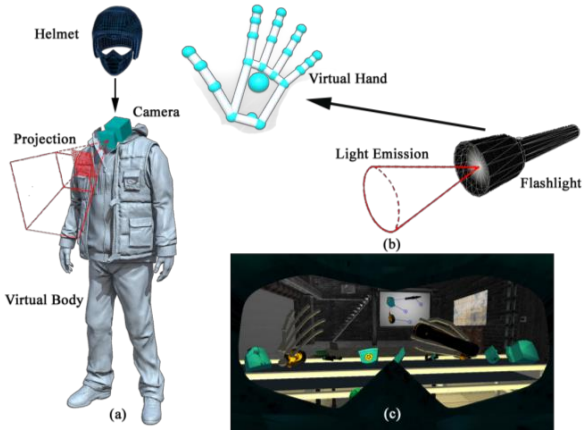

Figure 7. Child/Parent relations based on Oculus sensors (a) and Leap sensors (b). Final scene projection (c)

In the previous section we discussed about object interaction based on the virtual hand system, provided by Leap sensors, as shown in Figure 7 (b). Now, we will present a method of linking the objects to the virtual camera, based on Oculus sensors. Assuming we have a helmet, presented as a part of the model assembly, users may attempt to put it on them, finding themself inside the object. The virtual camera is now covered with the 3D model, which block a part of the eye visualization. Based on the headset's rotation on all the three axis, the object is now repeating the translation and orientation transformations after the camera's point of view, see Figure 7 (c).

VR headset orientation is provided by the Oculus library, in the right-handed cartesian coordinate system, stored in a quaternion. In order to use the same transformations technique as presented in Algorithm 1, we have to convert data orientation in Euler angles, which store all the X, Y and Z rotation angles in a vector, Equations 6, 7 and 8, where q_r, q_x, q_y, q_z represent the four quaternion elements and ϕ, θ, ψ are the yaw, pitch and roll angle rotations in radians [17].

$$\phi = arctg\left(2 * \frac{(q_w q_x + q_y q_z)}{1 - 2({q_x}^2 + {q_y}^2)}\right) \quad (6)$$

$$\theta = arcsin\left(2 * (q_w q_y - q_z q_x)\right) \quad (7)$$

$$\psi = arctg\left(2 * \frac{(q_w q_z + q_x q_y)}{1 - 2({q_y}^2 + {q_z}^2)}\right) \quad (8)$$

The results are used to define helmet's correct orientation, then used together with camera's position in virtual world, we can render the 3D model in front of the camera.

Algorithm 2. Head-object tracking transformation

```
GETOBJECTLINKEDTOCAMERA()
.IDENTITYMATRIX()
.TRANSLATION(camera.postion())
.ROTATION(camera.rotation().pitch, AXIS(1,0,0))
.ROTATION(camera.rotation().yaw, AXIS(0,1,0))
.ROTATION(camera.rotation().roll, AXIS(0,0,1))
.TRANSLATION(objectPositionOffset)
.SCALING(obj.scale())
.RENDERMODEL()
```

Naturally, light propagation or objects attached to other objects are following direct motion of the parent's model. For instance, placing a part of the assembly model in a pocket, will change its position based on the movement of the body. A light source emitted from a flashlight, which is hold by one of the hand has 2 levels of ascendancy.

MODEL BUILD

Model composition has steps and sometimes it has alternatives of the assembly process, by starting with base parts and finishing with covers, circuit boxes or supporting parts. The model building might be completed using parallel assembly, presented as a generic tree graph, which distributes the product parts in levels of requirements, Figure 8. The nodes represents the model parts and the connected networks define the attaching rules. Performing a level order traversal, we are making sure the correct workflow is provided.

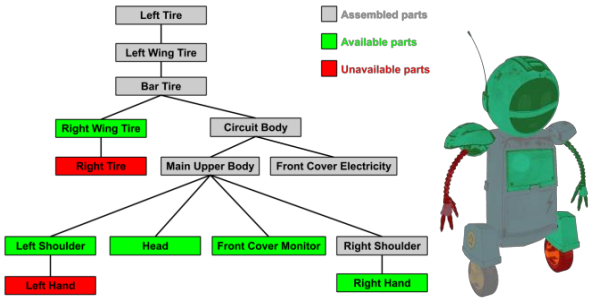

Figure 8. Assembly process based on N-array tree graph of the model parts

Creating an efficient mode of interaction with the objects, allow the user to stay still with the body. Model parts are split in hidden and visible objects, placed on a conveyor belt. Moving the belt forward or backward, will bring parts closer to the user and will make other parts visible to the camera's point of view. The assembly is constraints to all components, which makes the final model to be accomplished with all parts pieced together, which were initially on the conveyor belt.

Model creation

The engineering model has to be decomposed in parts and exported, using modelling tools, in the origin of the coordinate system. Based on object position and orientation in the final assembly form, the application can create events and correct areas of reconstruction in the virtual 3D world. All model parts are placed randomly on the conveyor belt and some rotation transformations are also applied to each of them. This way, the simulation will always be different. As not all the objects fit on the belt, some of them are excluded from the rendering process. Searching for other parts is done by sliding the belt in a certain direction, using directional buttons.

GUIDED ASSEMBLY

The application comes in many forms of learning, by evaluating the assembly performance, inspecting model parts, practice or testing the user composition capacity and problem solving skills. An instruction manual projected on a table might help people to identify the correct objects and accurately connect parts together, as shown in Figure 9. Novice users achieve experience by guided visual steps. All the possible parts are marked on screen, by creating a virtual bounding box model around the object, placed on conveyor belt. Grabbing the object will display the correct position and orientation on the assembly station.

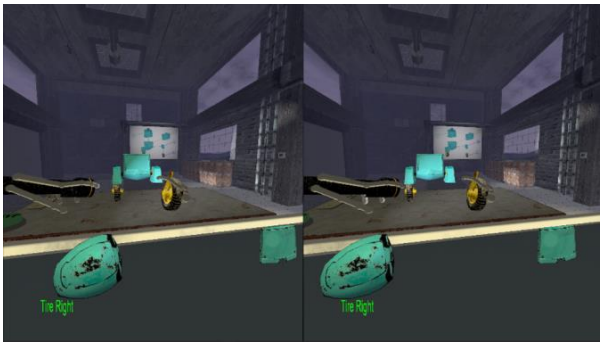

Figure 9. Virtual scene application split in two, corresponding to left and right eye

The Oriented Bounding Box (OOB) geometry mesh is computed using Vertex Buffer Objects (VBOs) by loading the lower and upper vertex limits of the object on all axis. The attributes are stored in memory as data vectors and therefore used to render the model behavior and properties on screen, like world position, texture coordinates, normals, colors, etc.

Some of the model parts might be similar, having the same structure, but different in terms of assembly steps. A labelling system is developed to show the object specifications, like name and metrics, so that users will be capable of associating easier and faster the parts, in order to achieve the final model. During the assembly session, users can check the parts properties from assembly station by passing the index finger through the model. Also this technique is used in triggering object events and buttons interaction.

PERFORMANCE

Performance is directly related with the human experience in using similar applications, motivation, attention to details and the level of knowledge. The first usage of virtual equipment by the user leads to accommodation with the system and adaptation with simple and complex interaction techniques available for use. In time, he might improve the experience, by training and by changing the application difficulty.

Other VR equipment uses physical buttons on hand controllers to activate interaction events. All buttons behavior are determined by the application and the interaction response might be defined as long-press or short-press of the button. Our solution set aside alternative touch interaction equipment, so that users are not learning new styles that are not related to real world interaction. Natural hand gestures detected by the LeapMotion sensor have the potential to be more natural and familiar than traditional methods. Based on hand gestures, the system analyzes the hand structure and performs one of the following actions: linking an object to the hand system, switching item's parent, releasing structure connection, running device response or enabling free object rotation. The amount of time spent by an user to understand the basic interaction features of the application is completed in a short period of time, achieved in the beginning of the simulation. To successfully make an action in the virtual world, the system leads the user intuition to try performing different gestures.

The assembly process requires users to have conceptual knowledge, analytical knowledge and metacognitive awareness. We did some tests on how much time it takes for the user to complete the reconstruction model of a robot composed of 14 parts, with both guided assembly methods combined and individuals. Five students and two employees aged between 21 and 35, who have not used a VR headset before, participated in this immersive assembly test. After some successful attempts, the following day, users were given the same model to assembly, without guidance elements. This way, we tested human retention and attentiveness. The final results are presented in Table 2.

Guidance	Both methods	Interactive Indicators	Instr. Manual	None
Completion Time (min.)	3:45	4:10	5:05	6:20

Table 2. Assembly performance time

SYSTEM REQUIREMENTS

The ideal virtual reality experience using DK2 is achieved with the maximum allowable latency of about 20ms and 75Hz application frame rate, based on the recommended hardware requirements for Oculus Rift. In this case, the whole scene is rendered in approximately 9ms and the rest of the time up to 13.33ms is allocated to image distortion for each frame. Once the double scene image is sent to headset screen, the system initializes the parameters for the next frame. The following software were used for the application implementation:

- Oculus Setup (latest version)
- Leap Motion Developer Kit version 4.0.0

The main libraries used are:

- Oculus SDK for Windows version 1.38.0
- Leap Motion Orion version 3.2.0

CONCLUSION

The assembly simulation using VR technology involves designing a model, breaking it down into pieces and developing a reconstruction process, based on linking rules. The purpose of this paper was to describe a virtual reality training tool of assembling a product in an immersive environment, based on human hand interaction. The engine developed processes the render algorithms, displays scene elements and computes 3D operations on objects based on user gestures. The overall solution inspired us to see many innovative ideas in which students and employees can benefit. Developing new learning tools by using current

technology may affect future education system, so that the new generation to be more advanced on performing practical skills.

Future improvements include further refinement of hand gestures and may involve using special gloves equipment with wireless technology for interaction with the virtual world. The sensation would be much similar with reality by feeling that the objects are held in the user's hand, based on tiny vibrations applied to the fingers. Combining soft interactions, such as hand gesture and hard interactions, such as vibro-tactile feedback provides more natural interaction with virtual objects, similar to manipulation tasks. The result allows an improvement in the sense of presence perceived by the user during the interaction.

REFERENCES

1. Active Virtual Reality Users Forecast Worldwide, Statistica Research Department. https://www.statista.com/statistics/426469/active-virtual-reality-users-worldwide
2. Daniel Kersten, Pascal Mamassian and David C Knill, Moving cast shadows induce apparent motion in depth, *Perception, vol. 26* (1997), 171–92.
3. Zhao, Richard & Aqlan, Faisal & Elliott, Lisa & Lum, Heather, Developing a virtual reality game for manufacturing education (2019).
4. Laia Pujol-Tost and M. Phil, Interactivity in virtual and multimedia environments: a meeting point for education and ICT in archaeological museums, *University of the Aegean*, Department of Cultural Technology and Communication (2020).
5. Paul Zimmons and Abigail Panter, The Influence of Rendering Quality on Presence and Task Performance in a Virtual Environment, *Proceedings - Virtual Reality Annual International Symposium* (2003), 293-294.
6. Mihai Pop and Adrian Sabou, Gesture-based Visual Analytics in Virtual Reality, *Revista Română de Interacţiune Om-Calculator 10, Issue 3* (2017), 216–230.
7. Thomas Galais, Rémy Alonso and Alexandra Delmas, Natural Interaction in Virtual Reality: Impact on the Cognitive Load, *The Human Factors and Ergonomics Society / Europe* (2019).
8. A.C. Boud, D.J. Haniff, C. Babe, and S.J. Steiner, Virtual reality and augmented reality as a training tool for assembly tasks, *Proceedings of IEEE International Conference on Information Visualization* (1999), 32–36.
9. Patricia Baggett and Andrzej Ehrenfeucht, Building physical and mental models in assembly tasks, *International Journal of Industrial Ergonomics, Volume 7, Issue 3* (1991), 217–227.
10. Unity, Unreal, Godot, How to choose the best real-time 3D solution. https://www.ausy.com/en/technical-news/unity-unreal-godothow-choose-best-real-time-3d-solution-0
11. Kevin Horowitz, Skeletal Tracking 101: Getting Started with the Bone API. http://blog.leapmotion.com/skeletal-tracking-101-getting-started-with-the-bone-api-and-rigged-hands
12. Alex Colgan, Hand Hierarchy. http://blog.leapmotion.com/getting-started-leap-motion-sdk/hand-hierarchy
13. Oculus Documentation, Rendering to the Oculus Rift. https://developer.oculus.com/documentation/native/pc/dg-render
14. A. D. Bradley, B. Karen and A. B. Phillips, Oculus Rift in Action, *Manning Publications* (2015).
15. Computing the Hand Orientation. https://developer-archive.leapmotion.com/documentation/csharp/devguide/Leap_Hand.html
16. Steven M. LaValle, Planning Algorithms, *Cambridge University Press* (2006), 97-100.
17. Josè Luis Blanco Claraco, A tutorial on SE(3) transformation parameterizations and on-manifold optimization, *University of Málaga* (2020), 15–16.