EPJ Web of Conferences **251**, 01006 (2021) https://doi.org/10.1051/epjconf/202125101006
CHEP 2021

Browser-based visualization framework Tracer for Outreach & Education

Alexander Sharmazanashvili[1*], *Nikoloz* Udzilauri[1], *Shota* Kobakhidze[1], *Luka* Todua[1], *Nino* Zurashvili[1], and *Irakli* Kverenchkhiladze[1]

[1]Georgian Technical University, Tbilisi, Georgia

Abstract. Education & outreach is an important part of HEP experiments. With outreach & education, experiments can have an impact on the public, students and their teachers, as well as policymakers and the media. The tools and methods for visualization enable to represent the detectors' facilities, explaining their purpose, functionalities, development histories, and participant institutes. In addition, they make it possible to visualize different physical events together with important parameters and plots for physics analyses. 3D visualization and advanced VR (Virtual Reality), AR (Augmented Reality) and MR (Mixed Reality) extensions are the keys for successful outreach & education. This paper describes requirements and methods for the creation of browser-based visualization applications for outreach & education. The visualization framework TRACER is considered as a case study.

1. Introduction

3D visualization aims of the graphical representation of the detector's facilities and particles interaction events. Methods for the 3D visualization enable to learn the structure of facilities and provide basic control like zooming, rotation, movement and learning the physical parameters like tracks particles, hits, Jets, missing energy, energy deposits into materials, etc.

Interaction with 3D models is possible by selection of the different components in the visualization scene, like facilities sub-elements, or tracks, etc. and getting descriptive information about the selected items. Another important functionality implemented by the 3D visualization method is geometry cuts. They enable to set different views of structures, like 1/2, 3/4, 5/4 cuts, etc. for better understanding of detectors and the event distribution topology inside the detectors (Fig. 1).

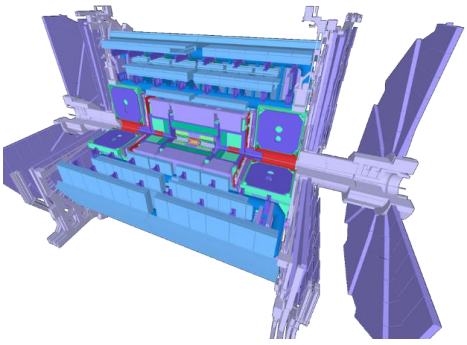

Fig. 1. Geometry cuts in the TRACER Framework

Augmented Reality (AR) visualization is a highly demanded technology for outreach and education in HEP experiments. There are several branches of AR implementations in HEP.

ART - Augmented Reality Table is a method, which enables to 'put' experiments on the discussion table by using portable devices, like telephones and tablets. ARD - Augmented Reality Door is a method for navigation inside of detectors by moving portable device together with an active camera. LND - Augmented Reality LaNDscape is a method for the visualization of detectors in actual scale in a real-life environment for a better imagination of the physical dimensions. ARB - Augmented Reality Book is a method for the development of extensions for paper-printed books, posters, flyers, etc. VR - Virtual Reality method based on a realistic representation of real-life scenes in 3D and enables to consider the observer as a part of it [1]. So, the observer feels like being inside the scene (fig.2) and can move through it using the different 'fly' modes. It is possible by using special headsets, audio facilities and advanced haptic tools [2]. The simplest representative of VR headset is Google cardboard [3]. It enables to put a portable

* Corresponding author: lasha.sharmazanashvili@cern.ch

device inside a cardboard and activate stereo mode on the device. The VR application splits the screen into two symmetrical parts and synchronizes the screen updates according to the movement of the device.

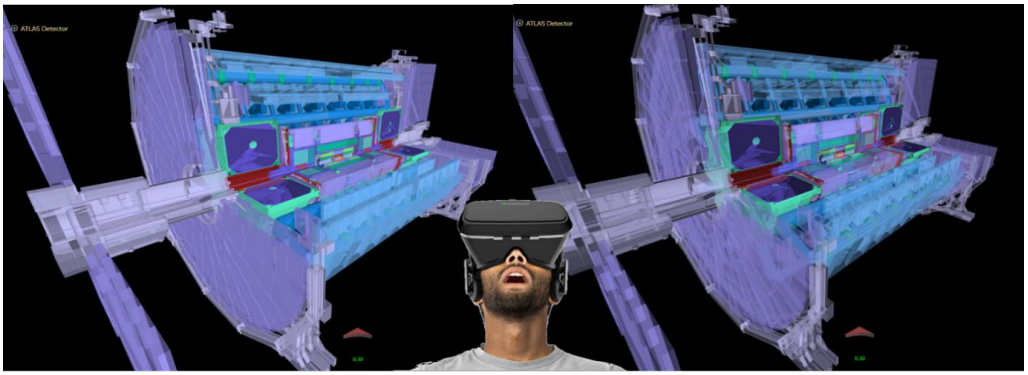

Fig. 2. TRACER application with a VR headset

The VR method visualizes only the digital content of the scene, while the AR method combines real and virtual scenes. Through the screen of portable devices, the digital scene is overlaying in the real-life environment. The mixture of the VR cardboard method with the AR method brings the so-called MR, Mixed Reality method. With the MR method the observer is again considered a part of the 3D scene through the use of a headsets. In addition to the real-life environment, applications are overlying the digital content in the scene and it is possible to interact with it by the using one's hands or additional haptic tools. Interaction foresees - basic control functions - zooming, rotation, movement, selections,

getting additional descriptive information, etc. The typical representative of a MR tool is Microsoft is Hololens [4].

2. Selection of a Visualization Platform

Visualization applications are built on base of the graphical engines. They are responsible for the graphical representation of the objects and scenes, rendering and functions for control. Graphical engines expedite the process of developing applications through existing templates and assets that can be reused, minimizing or completely extinguishing the need to have a deep knowledge of programming [5]. Moreover, graphical engines provide a chance to develop an application once and port it to various platforms, including portable devices, by making just a few changes to the original version.

Nowadays graphical engines are rapidly developing for the gaming industry. Often, they are called gaming-engines. For the moment there are >500 existing gaming engines. They cover a wide range of different user profiles and needs. Most of them are open sources with a large number of developers. Thus, the gaming-engine industry is fast growing and engines itself implementing more and more advanced methods of the visualization including AR, VR, and MR. However, there is not a single game engine that is best for all purposes.

Therefore, the use of gaming engines for HEP outreach & education can bring a big advantage to HEP applications. However, at the same time, choosing the right gaming engine is a subject for a precise analysis and investigation because they have limited possibilities. HEP scenes for visualization are heavy and complex. Therefore, implementation of gaming engines requires the development of special methods and tools for visualization in order to find agreement between the limitations coming from the gaming engines and the requirements of HEP visualization applications.

The main requirement coming from the fact that HEP visualization applications aim to serve a large audience with limited skills in IT or other professional skills. HEP is organizing wide events, like the Open days, OpenLabs cognitive festivals around the world [7]; OpenData sessions for spreading the research data in the educational branches - schools, universities, centers; International Particle Physics Masterclasses (IPPOG) [6] - 'Hands on Particle Physics'. IPPOG masterclasses are for 15-20 years old teenagers for helping them to choose the path in future professional careers. 200 universities from the 52 countries organizing the masterclasses for >13'000 students per year. Visualization applications are playing an important role in those events enabling virtual tours and detector event displays.

Also, HEP research is always related to a large number of workshops, conferences, scientific meetings. All these events use visualization applications for scientific data representation and cognitive activity - digital extensions of posters and other paper materials, books, etc.

Thus, visualization applications should be extensive, easily accessible, and compatible with the majority of the hardware and the operating systems, simple in use, with a well-developed user framework.

More than 500 gaming engines were investigated for the purpose of their implementation in HEP outreach & education applications. It was concluded that the best fit for HEP visualization requirements, is given by browser-based gaming engines and applications developed on their base.

Applications themselves have to build on the PHP/Javascript platform and interpreted online by built-in interpretators of the browsers. Therefore, there is no binary codes of the application. Applications will run on client devices without any installations. Another fit with the HEP visualization requirements is compatibility. Because the applications are running inside browsers and at the same time, browsers are compatible with a majority of hardware and software platforms, applications will derive the same benefit. Applications will also be easily reachable. Typing the URL address in the browser window will be enough to run the application.

On the other hand, the fact that the application is running inside a browser will bring limitations in the performance in comparison to binary applications. The performance also will depend on the accessible bandwidth because all data is downloaded from a server.

Almost all browsers are using the WebGL (Web Graphics Library) API (Application Programming Interface) for the rendering of interactive 2D and 3D graphical scenes [8]. WebGL is a JavaScript API based on OpenGL. There are several WebGL JS libraries for 3D visualization in browsers – Babylon.js, three.js, Phaser.js, etc. Their usage brings huge benefits, cause they are open source and fast-growing platforms. They deliver powerful features of WebGL-based and physical-based 3D rendering [9], directional-point-spot lighting, static and skinned meshes, texturing, importing of geometries, etc.

Thus, the best fit for HEP outreach & education requirements are browser-based visualization applications built on the WebGL graphical engine. The main difficulties, in building such kinds of applications are ensuring high performance and high quality of the 3D scenes.

3. Application Development

Application performance and scene quality are in direct dependence on the geometries visualized by the application. The ATLAS detector has very complex geometry and it is impossible to visualize it by WebGL built-in methods. Geometries have to be imported in facet-based formats through sets of triangles. However, all engines have limitations in the number of triangles to be represented in a scene simultaneously. For WebGL, it is about *two* million triangles in the scene. Above this number, application performance is either too low (*fps*=1-2) or browser start killing the process. An analysis shows that the full imported geometry of the ATLAS detector has about *thirty* million triangles, which is 15 times above the WebGL limitations. Therefore, geometries have to be simplified for the minimal number of triangles and at the same time, a high quality of scenes should be provided. Reduction of triangles brings increasing levels of approximation, which lowers the quality of representation. Another task is the reorganization of the topology of geometries - reorder the sequences of the volume descriptions for high- performance loading.

Loaders, software modules, responsible for downloading data from servers and uploading in the 3D scenes have an important influence on the performance. Loader algorithms have to be a well optimized for parallel and 'intuitive' loading, for the best management of the cash memory, etc.

The visualization applications for HEP outreach & education have different purposes and implementations:

- Event Displays are applications for the visualization of physical events inside detectors. The applications read reconstructed data, usually in the XML format, and visualize the different characteristics of events, like particle tracks, jets, etc. It interprets the distribution of particles in the detector components, energy deposits into the materials, the influence of the magnetic fields on the movement path, etc.
- Detector Displays are applications for the visualization of the structure, topology and subcomponents of HEP detectors/experiments. The applications give detailed explanations of the components, their connectivity, and purpose, used technologies, and the construction histories.
- VR, AR, MR extensions are a group of applications for the interaction with detectors, virtual touring and various cognitive extensions.
- Applications for Physics Masterclasses enable the organization of training courses in particle physics. They are covering various exercises of classwork and theoretical explanations about the physics, detectors, and used technologies.

All these applications are overlapping in on their basic functions. They are using common geometry descriptions, basic controls of the scene and have common inputs and outputs. In most cases, they built on the same graphical engines and have a very similar graphical user interface. Therefore, there is no purpose to repeat developments between applications. It is better to have a 'parent-child' architecture. In this architecture, there is a core application with the basic functionality, common for all child applications. The core does not express any specific purpose of the applications

mentioned above. Special purposes will be covered by child applications, so-called super systems of the core. Basic functionality will be derived to the super systems from the core.

This architecture will ensure a fast and flexible way for developing visualization applications.

4. TRACER Framework

TRACER is a 3-Dimensional visualization framework of the ATLAS detector developed by Nuclear Engineering Center at Georgian Technical University, Tbilisi, Georgia. TRACER is built on the WebGL/three.js engine and running inside browsers [10] https://tracer.web.cern.ch . This is an application written in Javascript/PHP and compatible to essentially all hardware devices - desktop PC, notebooks, tablets, and mobile phones. It runs on a majority of software platforms - MacOS, Win, Linux, Android, and iOS. The application does not require any installation on the client platform and follows the 'Click-and-Go' concept.

The Tracer architecture consists of a core part and several super system applications as illustrated in Fig.4. The core module is a set of basic functionalities yet never express as the specific tasks of the application. The core module is responsable for common functionalities like:

1. WebGL/three.js engine functioning, ensuring high performance
2. Geometry descriptions of the ATLAS detector
3. Geometry cuts
4. User interface

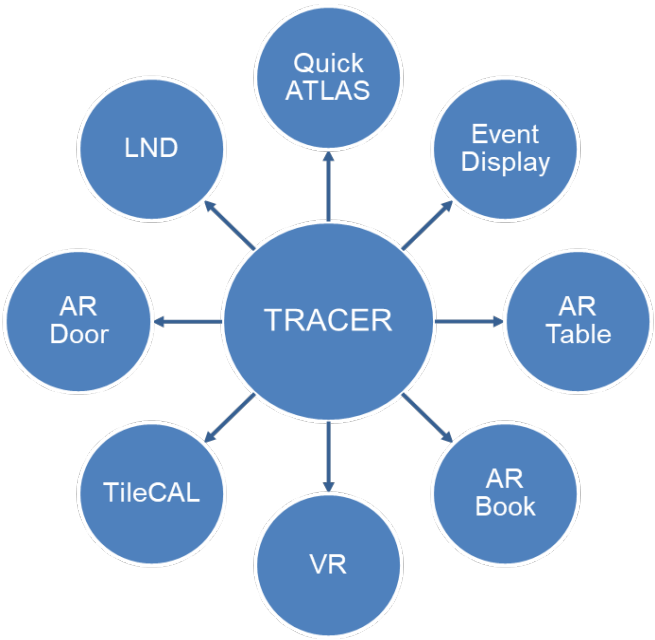

Fig. 3. Architecture of the TRACER

The Tracer core is built in pure Javascript, not using any JS subsets like typescript or AngularScript. As a result, the browser operates with a human-made native script and ensures high performance of the engine. Basic engine functions include the highly efficient and fast object loaders. For the downloaded detector component in the scene, the core basic modules automatically activate the download of the cut samples of the component in the local cache in background mode, with a preliminary assumption that the user can generate a request about geometry cuts. Then cuts will appear in the scene directly from the cache. All downloaded objects are stored in the local cache and ensuring fast appearance in the scene in case of their repeated call. Basic functions include also event loaders with their temporary storage in the local cache; Save/Open of the 3D scene configuration and presets; Object transparency; Grid, for the representation of objects in the metric system; proton-proton animation; dynamical distribution of particles inside the detector components; renderer with the optimized lighting and shadings.

The TRACER has a library with 246 geometry models, specially designed for browser-based visualization applications and ensuring a minimal amount of triangles and high quality of the 3D scenes. The special topology of the

geometry volumes brings the high performance of their visualization. All geometries are structured into 6 categories and 25 components. They have been simplified from the different sources and in total represented by the less than *one* million triangles, which is halth of the WebGL limitation.

Detector geometry structure visualized through the expendable hierarchical tree where elements and sub-elements are structured according to detector structure. This tree is an interactive part of the visualization scene and enables control to hide/unhide parts of the structural components. Thus, it makes it possible to visualize the detector in several configurations. Each component has separate visualization properties like contrast and transparency accessible through the selection. They are visible in the hierarchical tree. Therefore, users can choose specific configurations of the detector with emphasized components. The selection of the components on the tree activates the download process from the server or from the cache memory.

For geometry cuts, TRACER core has the *three* samples for each component - 3/4 cut, 1/2 longitudinal and 1/2 face cuts. In addition, TRACER core has a 5/4 cut which is the combination of the 3/4 + 1/2 face cuts. In this case both samples are downloaded in the scene.

As for the subsystems, TracerEVD, event display, and TracerVR, virtual reality applications are described below.

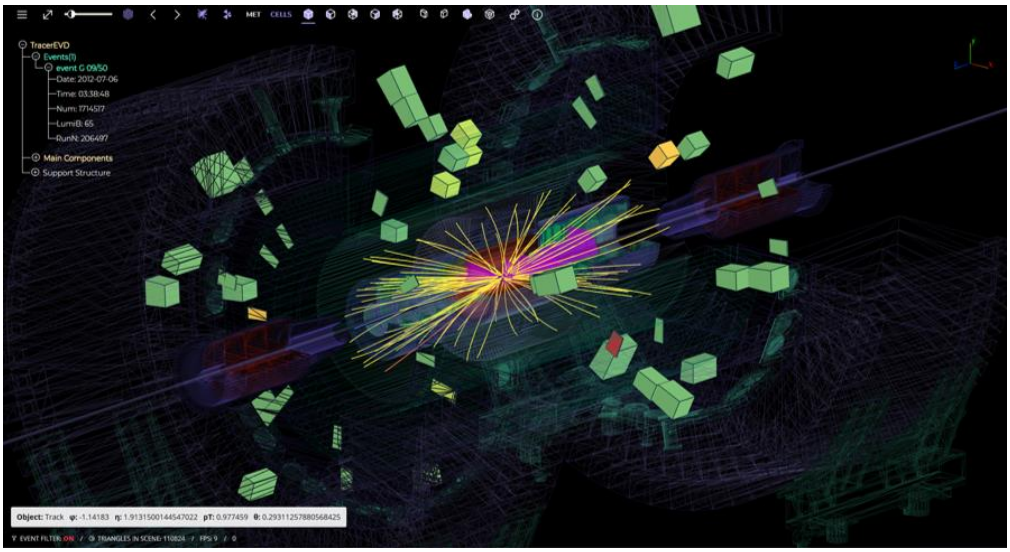

Fig. 4. Screenshot of the TracerEVD application

TracerEVD is a 3D browser-based event display [10] for the visualization of the ATLAS events (fig.4). The application is reachable from the TRACER framework https://tracer.web.cern.ch . TracerEVD visualizes particle tracks under the 6 different algorithms; jets under the 9 different algorithms; clusters; hits; energy deposits in the calorimeter cells and missing energy. All items are selectable by click and detailed information is appearing in a separate window. The application supports various filters by parameters like p_T, η, ϕ, θ, and E_T, particle types - Muon, electrons. The hierarchical tree shows the events in the scene. The main parameters of the event, like event name, date, time, number, luminosity block and run number, are presented in the tree. Events appear on top of the geometry of the detector subsystems. The geometry can be represented in different cuts, either in wireframe, or solid with adjustable transparency. Therefore, it makes it possible to set up high-quality scenes with a clear representation of the event topology inside the detector subsystems. TracerEVD visualizes events from the standard JiveXML files, uploaded by users or from 900 pre-setted events, prepared by the ATLAS open data.

TracerVR is a 3D browser-based Virtual Reality application for the organization of virtual visits to ATLAS Cavern (fig.5). The application is available on the TRACER framework https://tracer.web.cern.ch . It does not require special hardware/software platforms or engines. The application runs inside a browsers, using the average power of mobile phones and cheap Google cardboards. There are two modes of running the application – the headset mode and the 3D navigation mode. In the headset mode the portable device screen is divided into two stereo images with synchronized gyroscopic control. There are several episodes of virtual tours, giving different scenarios with various configurations of the detector. For instance, the inner detector episode, with a representation of Pixel, SCT, and TRT components inside the cavern, with accompanying information about the subsystems and physical events inside. Similar episodes are being prepared for the support structures, magnets, muon spectrometers, and calorimeters.

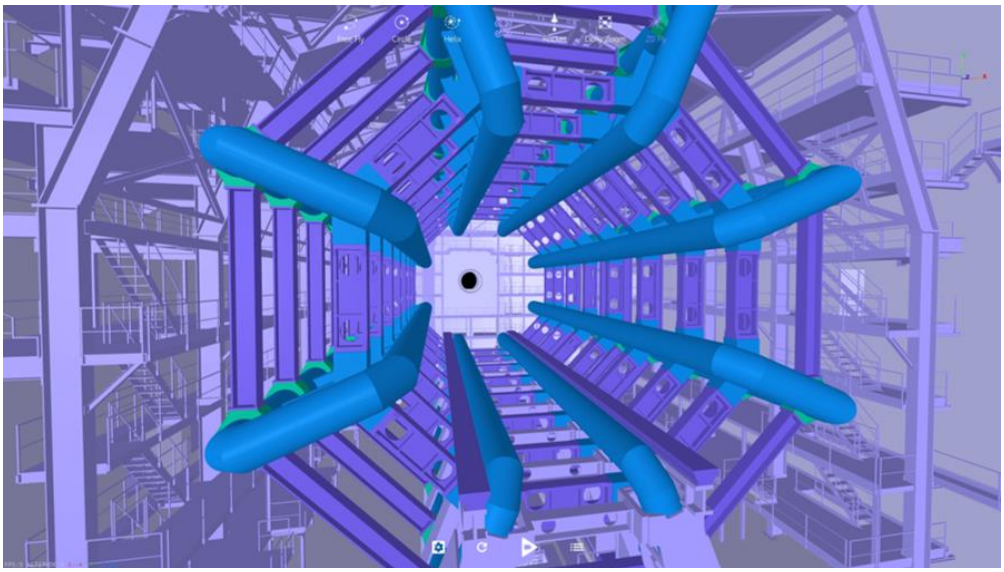

Fig. 5. Screenshot of the TracerVR application

In navigation mode, the virtual scene sticks with the screen of the portable devices carrying out Orbit control by touching the screen. So-called *'Drone'* flying functions enable easy and comfortable navigation inside the cavern:

- *'Free fly'*, with full control by the users

- *'Circle'*, fly on the circular path according to preliminary selected radius and altitude

- *'Helix'*, fly on an evolvent path with smoothly growing radius and altitude after each turn

- *'Rocket'*, vertical movement on top of the Z_0 beam intersection point on the preliminarily defined altitude

- *'Dolly Zoom'*, cinematic effect emulating forward/backward fly with proportional ZoomIn/ZoomOut control

- *'Z0 fly'* brings drone alongside the beamline with several rotations into Z_0 point.

5. Implementation

The Tracer visualization framework is going to play an important role in the Outreach & Education activity of the HEP experiments. Currently, Tracer is implemented by the IPPOG - International Particle Physics Outreach Group collaboration. IPPOG organizing annual sessions of masterclasses in particle physics. This is a one-day event for 15-18 years old teenagers setting the main objective, to provoke the interest in teenagers in the scientific life and research. IPPOG masterclasses methodology foresee activity which let listeners feel like be a HEP physicist for one day. Therefore, it is largely dependent on the visualization tools, to explain the basic structure and purpose of the detector, organizing the virtual tours, and learn experiments facilities, interpret and visualize data coming from the detector.

There is a positive experience of implementation of several subsystems of Tracer in the IPPOG masterclasses organized in Georgia in 2019, 2020, and 2021. The Tracer/QuckATLAS was using as a gaming tool, describing through the gaming scenario the basic structure of the ATLAS detector. The Tracer/VR was using for the virtual touring at Point 1. The Tracer/EVD was using as an event display application for collecting and interpreting the data in the masterclass exercises.

6. Conclusions

1. 3D visualization applications with advanced AR, MR and VR methods play a key role in the outreach & education activities of HEP experiments.
2. Usage of gaming engines will bring a unique advantage to the outreach & education visualization applications. However, to make it possible, new methods should be developed to find agreement between the limitations of gaming engines and HEP requirements.

3. Main requirements coming from outreach & education are that applications should be extensive, easily accessible, and compatible with the majority of hardware and software platforms, simple in use, with a well-developed user framework.
4. The best fit to HEP visualization requirements is given by browser-based gaming engines and applications developed on their basis. However, the major factors that define their usability are performance and quality of the visualization scene
5. Browser-based visualization applications should have a parent-child flexible architecture with a 'parent' part with core functionalities and a number of 'child' super systems executing the user-specific tasks.

References

1. R. D. Gandhi et al. "Virtual Reality - Opportunities and Challenges" / International Research Journal of Engineering and Technology, vol. **5**, Issue 1, (2018)
2. D. Wang et al. "Heptic Display for Virtual Reality: Progress and Challenges" / Virtual Reality & Intelligent Hardware, vol. **1**, Issue 2, p. 136-162 (2019)
3. A. Fabola et. Al. "Exploring the Past with Google Cardboard" / Digital Heritage International Congress (28 September - 2 October 2015) Granada, Spain, (2015)
4. Karthika S. et al. "Hololens" / International Journal of Computer Science and Mobile Computing, ISSN 2320-088X, Vol. **6** Issue 2, p.41-50 (2017)
5. E. Christopoilou et al. "Overview and Comperative Analyses of Game Engines for Desktop and Mobile Devices" / International Journal of Serious Games, issn: 2384-8766, vol.**4**, issue 4, (2017)
6. H. P. Beck et al. "Outreach Globally" / International Particle Physics Outreach Group, Report to ECFA, (2017)
7. Andrew Purcell et al. "CERN OpenLab Annual Report 2018" / Zenodo - Open-access Repository, (2019) https://zenodo.org/record3234404#.Xspt7sCYNaS
8. M. Bal et al, "Using WebGL in Developing Interactive Virtual Laboratories for Distance Engineering Education" / American Society for Engineering Education, (2017)
9. J. Jezek et al. *Design and Evolution of WebGL-Base Heat Map Visualization for Big Point Data* / Springer International Publishing AG (2017), DOI 10.1007/978-3-319-45123-7-2, (2017)
10. R.M. Bianchi et al. *Event Visualization in ATLAS* / ATL-SOFT-PROC-2017-046, (2017)