PROCEEDINGS OF SCIENCE

Development of the Browser-based 3D Visualisation Approach for the ATLAS Outreach Applications

Alexander Sharmazanashvili[a], Nino Zurashvili[a], Vladimir Dolinski[a], Mariam Nozadze[a]

a Georgian Technical University,
Tbilisi, Georgia

E-mail: lasha.sharmazanashvili@cern.ch

Outreach & Education is an essential part of High Energy Physics (HEP) experiments where visualisation is one of the key factor. 3D visualisation and advanced VR (Virtual Reality), AR (Augmented Reality), and MR (Mixed Reality) extensions make it possible to visualise detectors' facilities, explain their purpose, and functionalities, and visualise different physical events together with essential parameters. The visualisation applications should be extensive, easily accessible, compatible with most hardware and operating systems, simple in use, and with a well-developed user framework and open source. The best fit to these requirements brings browser-based applications based on the gaming engines. However, it causes limitations in the performance because codes are interpreted in real-time by the browsers and all data should be downloaded from the servers. Geometry descriptions play a critical role in finding agreement between the browser-based applications' performance and the quality of the visualization scene. Best cognitive results are delivered by so-called 'as-built' geometry descriptions. However, 'as-built' geometry is complex and consists of a decade of millions of primitives, parts and large assemblies. Therefore, bringing them into one visualization scene is almost impossible due to the engine's limitations. The paper describes the TRACER framework and 3D scenes development methods for AR/VR applications.

42nd International Conference on High Energy Physics (ICHEP2024)
18-24 July 2024
Prague, Czech Republic

1. Requirements

The browser-based visualization applications respond well to the outreach & education requirements – to be usable for a large audience with limited technical skills, be compatible with the majority of the hardware and software platforms and be **i**nstallation-free. A key factor for development is the implementation of gaming engines.

Nowadays, graphical engines are rapidly developing for the needs of the gaming industry. They cover a wide range of different user profiles and needs and are open sources with a large number of developers. They implement advanced visualisation methods, including AR, VR, and MR. However, gaming engines have limited possibilities because they are developed for particular game scenes and scenarios. HEP scenes for visualisation are much heavier and more complex. Therefore, the implementation of gaming engines requires the development of unique methods and tools for visualisation to find agreement between the limitations coming from the gaming engines and the performance of the visualisation applications.

The gaming engines have limited possibilities for the representation of the complex scenes of both types. For instance, the WebGL three.js engine by internal methods can represent geometry with up to 4’000 primitives inside. Scenes with geometries above this value make the visualization applications very slow, and useless for control and interaction. For the imported geometries the number of triangles in the scene should be less than 3 million. Otherwise, the browser kills the process because it is very slow [1].

2. TRACER Application

There are several visualization applications for outreach & education with different purposes and implementations – event display for visualization of the physical events, detector display for visualization of the detector hardware, VR/AR/MR applications for organization of virtual touring and interaction with facilities, applications for masterclasses, etc.

All these applications overlap in essential functions. They use the scene’s common geometry descriptions and basic controls and have standard inputs and outputs. In most cases, they are built on the base of the same graphical engines and have a similar graphical user interface. Therefore, there is no purpose in repeating the developments from one application to another. It is better to have a ‘parent-child’ architecture. This architecture has a core application with the basic functionality standard for all child applications. Fig.1 presents the architecture of the browser-based visualization application TRACER developed by the Nuclear Engineering centre at Georgian Technical University, Tbilisi, Georgia [2].

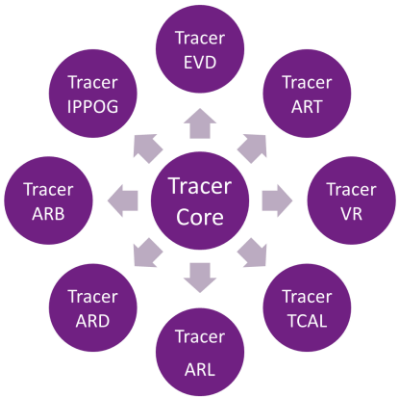

Fig.1 Architecture of the TRACER

The TRACER core uses React and TypeScript, which brings significant benefits. React's component-based architecture enhances code reusability and maintainability, while TypeScript's

type safety ensures reliable code. This combination improves productivity with features like auto-completion and refactoring support [3]. Moreover, the extensive React and TypeScript communities provide access to valuable resources. With these technologies, the TRACER core achieves optimal performance, due to highly efficient and fast object loaders. When a detector component is downloaded in the scene, the core module automatically activates the download of the cut samples of the component in the local cache in background mode, assuming that the user might request geometry cuts. This allows cuts to appear in the scene directly from the cache. All downloaded objects are stored in the local cache, ensuring fast loading in case of repeat calls. Other basic functions include event loaders with temporary storage in the local cache, the ability to save and open 3D scene configurations and preset, object transparency, a grid for representing objects in the metric system, proton-proton animation, dynamic particle distribution inside detector components, a renderer with optimised lighting and shadings.

The TRACER uses WebGL/three.js gaming engine for rendering of interactive 3D, and 2D graphics. The three.js provides a higher-level abstraction on top of WebGL, making it easier to create complex 3D scenes and animations, while WebGL handles the low-level rendering.

The TRACER Framework includes several specialized applications:

1. EVD (Event Display Application) - provides a 3D view of particle collision events in the ATLAS detector
2. TCAL (Tile Calorimeter) - offers an interactive 3D display of the Tile Calorimeter
3. ART (Augmented Reality Table) - brings the ATLAS detector to life on a discussion table using AR technology
4. ARB (Augmented Reality Book) - extends paper documents into 3D visualizations
5. VR (Virtual Reality Application) - delivers a realistic VR experience of the ATLAS Detector
6. EVD MC (Event Display for the International Masterclasses) - tailored for the IPPOG Masterclasses to display event data
7. ARD (Augmented Reality Door) - augmented reality door for navigation inside of the ATLAS Detector
8. ARL (Augmented Reality Landscape) - augmented reality landscape for visualizing the ATLAS Detector in its real scale and environment.

3. 3D Scene Development

The TRACER is compatible with the A-Frame which enables the creation [5] of captivating AR experiences on the web. Considering the different hosting options, SAAS (Software-as-a-Service) platforms offer the advantage of easy setup and management, with built-in features and tools specifically designed for AR development. However, they may have limitations in terms of customization and scalability compared to IaaS (Infrastructure-as-a-Service) or PaaS (Platform-as-a-Service) solutions. On the other hand, IaaS and PaaS platforms provide greater control and scalability, allowing for custom configurations and accommodating high traffic demands. However, they may require more technical expertise and setup time.

Also, it is important to note that geometry descriptions of the 3D scenes play a crucial role in creating AR experiences. 3D scenes are digital representations of real-world objects or characters, which can be overlaid on top of the real-world environment in AR. There are several software tools available for creating 3D scenes for AR. Some of the most popular options include 3D tools such as Blender, Maya, and 3ds Max, which are widely used for creating high-quality 3D scenes.

To optimize the performance of the application it is crucial to consider the following guidelines:

Geometry: To maintain a smooth and efficient experience, limiting the polygon count of 3D assets to below 100,000 polygons is recommended. This helps reduce memory usage and ensures faster rendering of complex models.

Draw Calls: Keeping the number of draw calls under 200 is important for GPU efficiency. When rendering 3D models in AR, each mesh or object within the scene requires a separate draw call to the GPU. Draw calls are essentially commands sent to the GPU to render a particular object or part of a scene. Excessive draw calls can increase rendering costs and degrade performance. The best number of draw calls for the visual/performance ratio is 200. By merging objects that share the same material and limiting the number of meshes in a USDZ/GLTF and other file formats to approximately 50, developers can minimize the overhead associated with draw calls.

Animations are an important part of the scene, but they increase draw calls, requiring an assessment of the pros and cons of each animation style. Skeletal animation is more realistic, but it requires more computational power. Morph target animation allows for flexible shape alterations at the cost of more processing. Object animation provides simplicity and economy at the expense of sophisticated control. Developers can create captivating AR experiences that optimize performance by carefully evaluating these elements [4].

4. Conclusion

1. Developing 3D browser-based TRACER applications requires careful consideration of tools and methods
2. Simplifying geometry and optimizing data management ensure smoother user experiences, leading to more accessible and impactful visualizations across various fields.

References

[1] Jose M. Noguera et al. “Visualization of Very Large 3D Volumes on Mobile Devices and WebGL”/ Proceedings of Conference WSCG Communication, January 2012

[2] Sharmazanashvili A. “Geometry Modelling in HEP” / Book ISBN 978-9941-8-5034-9 First edition, 2022, 504p

[3] Jordan Jaramillo “Why Should You Prefer TypeScript over JavaScript in React.JS?”/ https://dev.to/jordandev/why-should-you-prefer-typescript-over-javascript-in-reactjs-2d1

[4] Apple Developers. “Adding Visual Effects in AR Quick Look and RealityKit”/ https://developer.apple.com/documentation/arkit/adding_visual_effects_in_ar_quick_look_and_realitykit

[5] Reilly Grant et al. “DeviceOrientation Event Specification”/ W3C Working Draft, 21 April 2023 https://www.w3.org/TR/orientation-event/