

DEVELOPMENT OF NEXT-GENERATION EVENT VISUALIZATION PLATFORM FOR THE BM@N EXPERIMENT

E. Blinova^a, I. Dunaev^b, K. Gertsenberger^c, P. Klimai^{d,b,*}, A. Nozik^{b,d}

*^a MIREA - Russian Technological University, 78 Vernadsky Avenue,
Moscow, 119454, Russia*

*^b Moscow Institute of Physics and Technology, 9 Institutskiy per., Dolgoprudny,
Moscow region, 141701, Russia*

*^c Joint Institute for Nuclear Research, 6 Joliot-Curie, Dubna, Moscow region,
141980, Russia*

*^d Institute for Nuclear Research, Russian Academy of Sciences, 60th October
Anniversary Prospect 7a, Moscow, 117312, Russia*

*e-mail: pklimai@gmail.com

Received November 12, 2023

Abstract – In high-energy physics experiments the ability to display both detector geometry and physical objects of particle collision events, such as hits and particle tracks has become an essential feature required for physicists to better understand particular collision events as well as to present the physical results to a wider audience. Currently, most experimental collaborations build their own event display solutions with little to no unification between them. In this work, a new event visualization solution for the BM@N (Baryonic Matter at Nuclotron) experiment, a fixed target experiment of the NICA (Nuclotron-based Ion Collider fAcility) project, is presented. The solution is based on VisionForge, a modern open-source visualization system. An important part of the solution is integration of the system with the experiment's software framework, BmnRoot, which is a CERN ROOT-based environment. Several possible methods of such integration are discussed and the established architecture of the next-generation visualization system is explained.

INTRODUCTION

In experimental high-energy physics the task of the visualization of both detector geometry and objects of particle collision events, such as clusters, hits, particle tracks and their points for registered and simulated physics events is essential. Such visualization helps to understand particular collision events better, illustrate the obtained results and present them to a wide audience. In this regard, the BM@N (Baryonic Matter at Nuclotron) experiment, a fixed target experiment of the NICA (Nuclotron-based Ion Collider fAcility) project [1], is not an exception. It requires a visualization system that is integrated with other experiment software systems [2, 3] such as the main framework of the BM@N experiment, BmnRoot. In particular, it must be able to properly extract the information stored in data files employing the CERN ROOT [4] format.

Multiple event visualization systems, often referred to as “event displays”, have been previously developed (see [5, 6] for a review), but no universal solution is available due to variety of experiment goals and used software systems and data formats. Thus, generally, to provide an event display solution, one typically has to choose some base software platform and adapt it to the purpose of the given experiment. For the present work, VisionForge [7] solution has been selected. In the next section, the main features of VisionForge are reviewed, and then its usage with BM@N software infrastructure is explained.

VISIONFORGE PLATFORM

VisionForge is an open-source platform for developing modern visualization solutions, implemented using Kotlin-multiplatform [8, 9] technology. A key feature of the platform is ability to create distributed dynamic systems: a rendering model can be created on one node (e.g., server), transferred to another node (e.g., client), and processed there. In addition, nodes can exchange updates to the data model without transferring the entire model each time.

VisionForge uses single, language-independent representation of the visualization model which enables the use of components and data sources written in different programming languages and embedded in different applications. The main structural unit of the VisionForge model is the view component called Vision. Each Vision may include immutable arguments that are passed when the view is created, as well as a dynamic metadata tree. Components of the metadata structure can be changed and their changes are automatically tracked by the framework and transmitted to other nodes in the form of a tree of values, which represents the update that must be applied to the initial tree. Simply put, if only one value has changed in a large data structure, only that value will be transferred during the update.

The VisionForge event model supports accumulation of multiple updates into a bigger batch and sending the accumulated updates. Thus, the platform enables building real time solutions – even if updates arrive faster than they can be sent to the visualization server, the user interface is not blocked.

INTEGRATION WITH THE BMNROOT ENVIRONMENT

The features listed make VisionForge well suitable for visualizing event data from accelerator experiments such as BM@N, where the total size of the visualization model can be very large, and transferring and rendering the entire model requires significant resources. Because with VisionForge one can transfer only part of the model, for example, only events

tracks, different events can be represented without redrawing the entire geometry, which gives a significant performance gain.

As VisionForge is not natively based on ROOT, its use for BM@N event display solution requires developing a software module transforming the ROOT data model into the VisionForge model. This task can be solved in various ways. One option is performing direct reading of the ROOT format. However, it is worth noting that the format stores not only the binary representation of the class instances, but also links (sequentially numbered references) to other objects, and those references depend on the object tree traversal order. A solution for reading serialized ROOT objects directly from a Kotlin program is a work currently in progress. For the present implementation of the visualization system, a JSON representation generated using TBufferJSON class methods (part of ROOT software distribution) was used. The JSON documents generated using this approach are provided to the visualization framework by the developed REST API server named “visapi” that executes the ROOT macros extracting geometry and track information as needed. A VisionForge server component (performing data model manager role) uses the obtained data to transform the detector geometry and particle tracks to the VisionForge representation model and pass them to the front-end visualization service. The overall architecture of the system is shown in Fig.1.

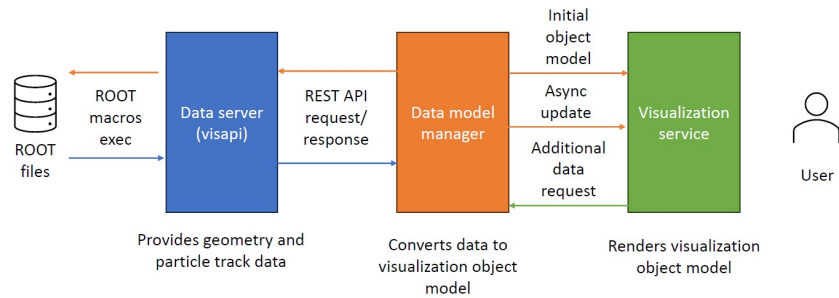


Fig. 1. Basic architecture and components of BM@N visualization system

The full BM@N geometry model includes more than 400,000 primitives, so rendering them all requires significant resources. In addition, constructing and displaying edges for many similar closely located elements (such as calorimeter cells) is not justified. To solve the problems, so-called “prototypes” have been implemented in the VisionForge model for three-dimensional objects. For a single prototype its geometry is rendered only once and reused for multiple objects, which helps to significantly reduce memory usage.

RESULTS AND DISCUSSION

Fig. 2 shows an example view of the developed Web interface of the system. The result of visualizing the BM@N geometry for given experiment runs (with the reference to the

geometry file being obtained by querying the condition database of the experiment), as well as the event data imported from the provided experimental file is presented. The user can conveniently move and rotate the view, zoom-in and zoom-out the scene as needed. In the right part of the window, the Settings workspace is shown, which enables exporting and importing the current canvas configuration (including object settings that are manually configured in the interface) to the JSON configuration file.

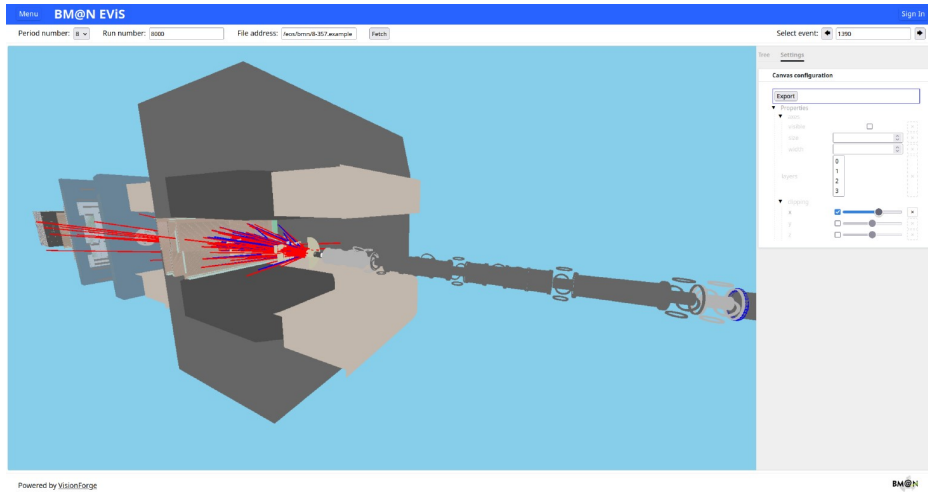


Fig. 2. Example view of the BM@N visualization system interface

In Fig. 3, another example view of the visualization system interface is shown, including scene graph (object tree). Fig. 3 also illustrates how for every object it is possible to change display properties such as color, opacity and visibility.

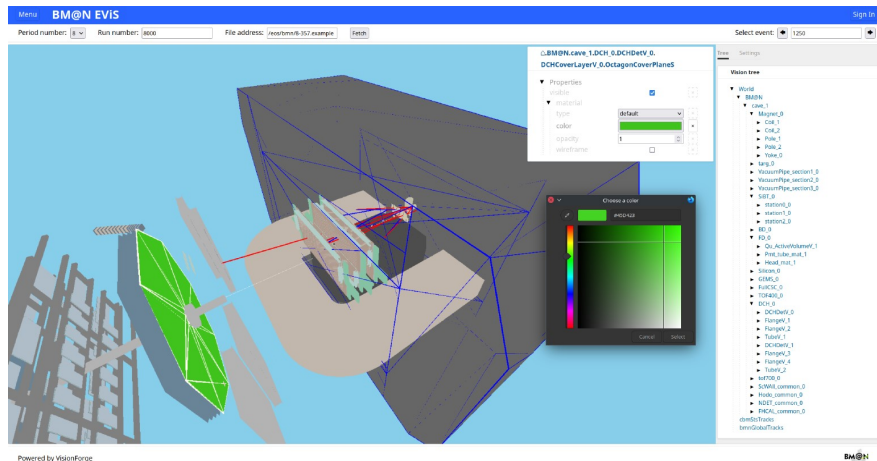


Fig. 3. Example of scene graph and object visualization tuning

It is worth noting that Kotlin-multiplatform technology, on which the developed system is based on, is available for Java Virtual Machine (JVM), JavaScript and native environments. As one of the practical applications, all visualization machinery, including ROOT data importing, can be compiled into a standalone HTML bundle file (with embedded JavaScript)

that is conveniently opened in a Web browser with no requirements for a running server application. Performing visualization using virtual reality (VR) is a natural next step possible with the platform, and is a work currently in progress.

CONCLUSIONS

A new visualization system for the BM@N experiment at NICA has been developed, providing necessary features of displaying detector geometry, particle tracks, and ability to control and fine tune the display via settings of any elements of the scene graph. The system is integrated with experiments CERN ROOT-based software environment, works on multiple platforms and is optimized for rendering geometries with hundreds of thousands of elements.

REFERENCES

1. *Blaschke D. et al.* [NICA Collaboration]. Searching for a QCD mixed phase at the Nuclotron-based Ion Collider fAcility (NICA White Paper). Dubna, JINR, 2014. 334 p.
2. *Batyuk P., Gertsenberger K., Merts S., Rogachevsky O.* The BmnRoot framework for experimental data processing in the BM@N experiment at NICA // EPJ Web of Conf. 2019. V. 214. P. 05027.
3. *Gertsenberger K., Alexandrov I., Filozova I., Alexandrov E., Moshkin A., Chebotov A., Mineev M., Pryahina D., Shestakova G., Yakovlev A., Nozik A., Klimai P.* Development of Information Systems for Online and Offline Data Processing in the NICA Experiments // Phys. Part. Nucl. 2021. V. 52. P. 801.
4. *Brun R., Rademakers F.* ROOT - An Object Oriented Data Analysis Framework // Nucl. Inst. & Meth. in Phys. Res. A. 1997. V. 389. P. 81.
5. *Bianchi R. M., Boudreau J., Konstantinidis N., Martyniuk A. C., Moyse E., Thomas J., Waugh B. M., Yallup D. P. on behalf of the ATLAS Collaboration.* Event visualization in ATLAS // J. Phys.: Conf. Ser. 2017. V. 898. P. 072014.
6. *Gertsenberger K.* Event Display for the Fixed Target Experiment BM@N // EPJ Web of Conf. 2016. V. 108. P. 02022.
7. VisionForge project, “VisionForge” [software], version 0.2.0, 2022. Available from <https://github.com/SciProgCentre/visionforge> [accessed 2023-11-03].
8. *Jemerov D., Isakova S.* Kotlin in Action. Manning Publications Company, 2017.
9. *Nozik A.* Kotlin language for science and Kmath library // AIP Conf. Proc. 2019. V. 2163. P. 040004.