A method of detector description transformation to Unity and its application in BESIII

Kai-Xuan Huang, Zhi-Jun Li, Zhen Qian, Jiang Zhu, Hao-Yuan Li, Yu-Mei Zhang, Sheng-Sen Sun, and Zheng-Yun You

1School of Physics, Sun Yat-sen University, Guangzhou 510275, China
2Sino-French Institute of Nuclear Engineering and Technology, Sun Yat-sen University, Zhuhai 519082, China
3Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China
4University of Chinese Academy of Sciences, Beijing, 100049, China

Detector and event visualization is an essential part in software of HEP experiments. Modern visualization technique and multimedia production platform, such as Unity, provides more fancy display effects and professional extensions for visualization in HEP experiments. In this work, a method of automatic detector description transformation is presented, which can convert the complicated HEP detector geometry from GDML in offline software to 3D modeling in Unity. The method has been successfully applied in the BESIII experiment, and can be further developed into applications such as event display, data monitoring or virtual reality. It has great potentials in detector design, offline software development, physics analysis and outreach for the next generation HEP experiments, as well as applications in nuclear techniques for industry.

Keywords: Detector description, Visualization, Unity, GDML, BESIII

I. INTRODUCTION

Detector description and visualization play important roles in various aspects throughout the life cycle of a High Energy Physics (HEP) experiment, including detector design, optimization, simulation, reconstruction, detector commissioning, monitoring, event display and physics analysis, as well as outreach and education. In HEP Software Foundation (HSF) community white paper [1] and the roadmap for HEP software and computing R&D for the 2020s [2], suggestions and guideline for visualization tools and techniques in future experiments have been specifically discussed, which focus on detector geometry visualization, event display and interactivity.

The detectors in HEP experiments are usually large-scale scientific apparatus with complicated geometry, which are composed of up to millions of detector units, such as the ATLAS [3] and CMS [4] detector at the Large Hadron Collider (LHC) [5]. The detector description is developed based on professional geometry toolkit, in which the most popular ones are Geometry Description Markup Language (GDML) [6] and Detector Description Toolkit for High Energy Physics (DD4hep) [7]. The detector geometry information in GDML or DD4hep format can be imported into the offline software of an experiment to provide consistent detector description for different applications [8, 9].

However, to display the complicated detectors in offline software becomes difficult, since the HEP software community does not have a common visualization tool that meets the requirements of visualization effects, speed, efficiency and portability. The ROOT software [10] and its EVE package [11] have been used in some experiments for detector visualization and event display, such as ALICE [12], BESIII [13] and JUNO [14, 15]. Although the ROOT based event display tools have advantages with its development in the offline software framework, their visualization effects are limited and it is also difficult to be implemented to platforms other than Linux.

In recent years, the popular video game engines from industry, such as Unity [16], have been used in HEP experiments for detector visualization and event display. Unity is a cross-platform engine for creation of games, architecture, video and animation. It supports more than 20 kinds of different platforms, including Windows, Linux, macOS, iOS and Android, which makes Unity the most popular application development platform on Apple Store and Google Play. The applications of Unity in ATLAS [3], BelleII [17] and JUNO [18] have achieved fancy visualization effects, which makes Unity a promising platform for visualization in future HEP experiments.

Despite of the advantages of great display effects and cross-platform support, a critical problem to use Unity for detector visualization is that Unity does not support the commonly used HEP detector description, GDML and DD4hep. The developers have to use the 3D modeling system in Unity to reconstruct the complicated detector from scratch again, which not only brings extra human work and maintenance problem, but also could make the detector description in Unity inconsistent with that in offline software.

In this paper, we present a method to automate the conversion of GDML detector description into Unity. The full detector geometry and its architecture, so long as described with GDML or ROOT, can be imported into Unity for automatic 3D detector modeling. The correlation between detector elements and identifiers is also preserved, which makes it convenient for further developments into event display or virtual reality (VR) applications. The method has been tested and validated with the BESIII detector description [8, 9].

The rest of this paper is structured as follows. In Sect. II, we introduce the detector description, visualization in HEP offline software and Unity, respectively. In Sect. III, the method of transformation from GDML to Unity and detector data flow is presented. Its application in BESIII is introduced.
in Sect. IV. And the further development of applications is discussed in Sect. V.

II. DETECTOR GEOMETRY AND VISUALIZATION

A. Detector description in HEP software

Detector description is an indispensable part in HEP offline software, which provides the detector geometry and status information for different applications, including simulation, reconstruction, calibration, event display and physics analysis. The popular HEP infrastructure software, such as Geant4 [19] or ROOT [10], has their own specific detector construction system. If the software developers define the detector geometry in these software respectively, there could be potential inconsistency between them. To solve this problem, some framework provides software-independent detector description, such as GDML and DD4hep, for detector related applications in HEP offline software.

GDML [6] is a detector description language based on XML (eXtensible Markup Language) [20]. It describes detector information through a set of tags and attributes in plain text format to provide persistent detector description for an experiment. Several HEP experiments, including BESIII [8, 9], PHENIX [21], LHCb [22–24] and JUNO [25], have used GDML to describe and optimize the detector geometry in conceptual design and offline software development [26–28].

DD4hep [7] is a software framework for providing a complete solution for full detector description, including geometry, materials, visualization, readout, alignment, and calibration, for the full experiment life cycle. It offers a consistent description through a single source of detector information for simulation, reconstruction, analysis, etc. DD4hep aims at applications in the next generation HEP experiments, such as CEPC [29], ILC [30], FCC [31] and STCF [32].

B. Visualization of detector and events

One important application of detector description is to visualize it so that the users can have a distinct view to better understand the HEP detectors. It is extremely important at the stage of detector conceptual design and detector commissioning. In combination with the event visualization, the event display provides a powerful tool to demonstrate the detector response to the particles, which plays an essential role in offline software development and physics analysis.

However, as part of traditional industry design, Computer Aided Design (CAD) [33] has been dominant in several stages of an HEP experiment, including detector design, construction and commissioning. There have always been difficulties in sharing the 3D modeling information between CAD and HEP offline software, since they belong to two quite different fields, industrial design and high energy physics, respectively.

In HEP experiments, physicists used to develop the detector description and event visualization tool in HEP offline software. In the offline framework, the developers can make full use of the detector geometry service and event data model to retrieve the corresponding information. The event display tools are usually based on the popular HEP software Geant4 or ROOT, whose visualization functions are friendly to HEP users and are naturally convenient for visualization software development. For example, the BESIII event display software is based on the infrastructure visualization function of ROOT. With the upgrade of ROOT and its EVE package [11], the development of event display tool becomes more convenient. Several recent HEP experiments, such as ALICE [12], CMS [4], JUNO [25] and Mu2e [34], have developed the event display software based on ROOT EVE.

However, due to the limited support and the open-source feature of ROOT, its visualization effects are not satisfactory to meet the diverse requirements of physicists. Most of ROOT applications are also limited to the Linux platform only. For better visualization and interactivity, several event display tools have been developed based on external visualization software. For instance, Atlantis [35, 36] and VP1 [37] are the two general-purpose event display tools in ATLAS. CMS has also developed several visualization systems, such as Fireworks [38] and SketchUp [39].

C. 3D modeling and visualization in Unity

Unity is a professional video and game production engine in industry, which supports more than 20 platforms. It is particularly popular for iOS and Android mobile app development, and has recently been used in HEP for scientific research and outreach. The Cross-platform Atlas Multimedia Educational Lab for Interactive Analysis (CAMELIA) [40, 41] is an event display software based on Unity for ATLAS. Fig. 1 shows the visualization of the ATLAS detector and a proton-proton collision event in CAMELIA. Another event display tool based on Unity, the Event Live Animation with Unity for Neutrino Analysis (ELAINA) [18] has also been developed in JUNO, as shown in Fig. 2.

Fig. 1. CAMELIA, the ATLAS event display tool based on Unity.

The visualization software based on Unity has the following advantages:
Fig. 2. ELAINA, the JUNO event display tool based on Unity.

- **Fancy visualization effects.** The Unity engine, as a professional 3D software, provides more detailed description of the objects and more fancy visualization effects, which makes it much more powerful than the traditional event display based on ROOT.

- **Cross-platform supports.** Thanks to the multi-platform supports of Unity, after the development by building models and functions in Unity, a project can be directly exported and deployed in different operating systems, including Windows, Linux, macOS, iOS, Android and network browsers. This feature makes the same visualization project available for all the users on different platforms to enjoy. It not only reduces the workload in project development, but also makes it easier for maintenance.

- **Extensibility.** Unity allows the project to be extended and further developed into VR [42] or Augment Reality (AR) [43] project, such as BelleII VR [44], which provides a quite different way for detector design, offline software development and physics analysis. It will also be very helpful to advertise the scientific projects to the public, especially the physics behind the large-scale scientific apparatus, and makes it a powerful tool in education and outreach, such as the Total Event Visualiser (TEV) of the CERN Media Lab [45]. The supports for various VR or AR devices have been integrated in Unity and will be continuously supported with the upgrade of Unity, so that the visualization project developers can focus on the software project itself, instead of different hardware supports.

Benefits from the prominent features of Unity, it has huge potential and boosts promising prospects in the development of visualization software. Not only can it be used for scientific research in particle and nuclear physics, but also has wide applications in nuclear techniques for industry.

### III. METHODOLOGIES

Although Unity has great advantages in developing visualization software, a lot of endeavor has to be made in its application to the large-scale scientific apparatus. For example, the HEP detectors are usually highly complicated with millions of components, which makes it difficult to build the 3D modeling of detector in Unity.

To develop the event display tool for HEP experiments, the detector description in Unity should be fully consistent with that in offline software, so that the display of tracks, hits, showers and display of the detector components match with each other. Since for an HEP experiment, the detector description usually already exists in the offline software, we propose a method to automatically convert the HEP detector description into 3D modeling in Unity.

#### A. Automatic detector geometry transformation

In HEP offline software, to avoid the inconsistency of detector description in data processing, it is essential that the framework can provide a single source of detector description for all applications, such as simulation, reconstruction, calibration, event display and data analysis. This idea is usually realized by developing automatic geometry transformation interfaces between different software. For an existing detector description with GDML, the detector construction in Geant4 and ROOT for a specific experiment can be automatically completed with the GDML-Geant4 and GDML-ROOT interfaces [8, 9], respectively. In this way, the detector geometry consistency among the Geant4-based simulation, the ROOT-based reconstruction, event display and data analysis are automatically guaranteed.

A similar idea can also be implemented in the 3D detector modeling of Unity. Due to the complexity of HEP detectors and the fact that HEP experiments usually already have a detailed detector description, for example, in GDML format, it will save a lot of human work in development and maintenance if a method of automatic detector transformation from GDML to Unity is available. As mentioned in Sect. II C, Unity provides fancy visualization effects and multiple extensions for further development. If this novel method can be realized and since it is a universal technique, all current and future HEP experiments can benefit from it by easily development of applications for detector visualization, event display and outreach.

#### B. Detector data conversion from GDML to Unity

In the industry market, there are tens of popular 3D file formats, however, none of them is commonly supported in both of HEP software and Unity. So we need to find a data flow path that starts from GDML and ends in Unity with a minimum steps of conversion.

FreeCAD [46] is a general-purpose parametric 3D computer-aided design and modeling software. It is free and open-source, available for users to extend the functionality. Since FreeCAD supports the import of geometry in Constructive Solid Geometry (CSG) format [47], an interface between GDML and FreeCAD is available to import the GDML format detector description [48]. On the other hand, FreeCAD
allows to export the modeling in several kinds of popular 3D formats, including STEP [49], BREP [50], VRML [51], etc.

In all these formats, STEP is chosen for further geometry transformation. Although STEP is still not a format that can be directly read in by Unity, it can be transformed into the FBX format [52] with the Pixyz Studio software [53]. Pixyz Studio is a CAD data preparation and optimization software, which supports more than 35 3D file formats. It provides tessellation algorithms, enabling the transformation of CAD data from industry-leading solutions into lightweight, optimized meshes, in which FBX is one of the 3D mesh formats that Unity supports. It is noteworthy that Pixyz has joined Unity so that Unity can provide plugin to import and transform heavy and complex 3D CAD data, such as HEP detectors, into optimized meshes for real-time 3D engines more conveniently.

The full chain of data flow is shown in Fig. 3. The original HEP detector description in GDML format usually comprises of the following parts: the material list, the position and rotation list, the shape list, the detector component (physical node) list and hierarchy of the whole detector tree. The GDML detector description is firstly imported into FreeCAD with the GDML-FreeCAD interface, then it is exported into the STEP format. Pixzy reads in the STEP format data and transforms it into FBX format, which can be directly read in by Unity, and keep the detector unit association information in the meantime.

![Fig. 3. Detector data flow from GDML to Unity.](image)

With the data flow chain described above, we provide a method to transform the GDML detector description into Unity with automated 3D detector construction. The method is realized with the GDML-CAD interface, FreeCAD and Pixyz software. With the 3D detector modeling successfully constructed in Unity with automation, further application development based on Unity will be achievable.

C. Integrity in Unity 3D modeling

For a specific HEP detector with GDML description, to realize a complete 3D modeling of the detector in Unity, a few more extra works are necessary, in addition to the automatic geometry transformation method introduced above.

First, in the GDML-FreeCAD interface [48], not all of the solid shapes that are currently defined in Geant4 or ROOT are fully supported. The interface only supports about 10 kinds of shapes, most of them are the basic CSG shapes. There are more than 30 kinds of specific shapes that are provided in Geant4 but not commonly used. For a specific HEP detector description with GDML format, if some special shapes are used but not available in the interface, users may request to develop the corresponding shape transformation in the GDML-FreeCAD interface, so that the full detector transformation can be supported.

Secondly, in description of a HEP detector, it is important to keep the association between a unique detector unit and its identifier in event data model. Such association information is essential to control the visualization properties of the detector unit in further application development in Unity such as event display. Usually, the identifier information is stored in the name of each node in GDML, so that later it can be extracted from the unit’s name in Unity to retrieve the mapping of a unit and its corresponding identifier. Hence, in transformation of the detector geometry from GDML to Unity, it is critical to retain the name of each detector unit to keep the association information.

Third, although the basic information such as density and makeup of materials can be transformed from GDML, Unity provides more rich visualization properties such as material color, texture, transparency, reflection, etc., which allows the user to have the freedom to render the detector geometry with more powerful visualization effects.

With the above supplementary information provided, the 3D detector modeling has been successfully constructed in Unity, and it has been qualified for further development.

IV. APPLICATION IN BESIII

A. BESIII detector description and visualization

BESIII is a spectrometer operating at the Beijing Electron Positron Collider II (BEPCII). The BESIII detector records symmetric $e^+e^-$ collisions provided by the BEPCII storage ring [54], which operates in the center-of-mass energy range from 2.0 to 4.7 GeV [55]. The cylindrical core of the BESIII detector covers 93% of the full solid angle and consists of a helium-based multi-layer drift chamber (MDC), a plastic scintillator time-of-flight system (TOF), and a CsI(Tl) electromagnetic calorimeter (EMC), which are all enclosed in a superconducting solenoidal magnet providing a 1.0 T magnetic field. The solenoid is supported by an octagonal flux-return yoke with resistive plate counter muon identification modules interleaved with steel (MUC).
The detector description in BESIII offline software is provided in GDML format. Each of the four sub-detectors, MDC, TOF, EMC and MUC, has a corresponding GDML file to describe it. A general GDML file provides description of the common materials and the other passive detector components, such as beam-pipe and superconducting solenoid. All applications in BESIII offline software, including simulation, reconstruction, event display, calibration and analysis, use the GDML files as single source of detector information.

BESIII Visualization (BesVis), an event display tool based on ROOT, has been developed to visualize the detector and to analyze the physics events, as shown in Fig. 4. It has played an important role in BESIII offline software development and physics analysis since 2005.

**Fig. 4.** Visualization of the BESIII detector and a real data event in BesVis.

### B. Conversion of BESIII geometry to Unity

Based on the method introduced in Sect.III, it is possible to perform a format conversion of the BESIII detectors description from GDML to Unity. The process of detector data conversion is divided into the following steps.

First, the GDML description of each sub-detector is imported into FreeCAD with the GDML-CAD interface. Since the original interface only supports very basic CSG shapes but BESIII has used some complicated shapes, such as Polygon, Polyhedra, Twistedtubs, IrregBox and Boolean shapes, we update the interface to allow correct conversion of all the shapes being used in BESIII GDML detector description.

Second, after importing into FreeCAD, the BESIII detector data is exported in STEP format. Then the STEP files are converted to FBX files with the format conversion function provided by Pixyz Studio. As an example, visualization of the BESIII TOF sub-detector in FreeCAD and Pixyz is shown in Fig. 5.

**Fig. 5.** Visualization of the BESIII TOF sub-detector in FreeCAD (left) and Pixyz (right).

Third, the FBX files can be directly imported into Unity. For the BESIII detector, there are four sub-detector FBX files (MDC, TOF, EMC and MUC) and one general FBX file to describe the beam-pipe and superconducting solenoid. The sub-detectors have been checked in Geant4 and ROOT to guarantee no space overlaps between each other, so the subdetectors can be directly combined together to form the whole detector. Finally, the 3D modeling of BESIII detector is successfully constructed in Unity with a set of automated steps.

### C. Display of BESIII detector in Unity

Although at this stage, the BESIII detector has been automatically built in Unity, it is not well displayed in the scene. Since GDML does not save the visualization attributes, all detector units are still set with the default visualization attributes in Unity. All that the users can see in the scene is a box, which is the top world volume in definition of the GDML detector description.

A set of scripts need to be developed to set color, transparency, reflectivity and texture of material for the detector units. The top world volume and the virtual mother volumes need to be set invisible, to allow the inner detector units to show up. Another advantage of the automatic conversion is that the name of each detector unit is kept during the process of transformation, so that the scripts can set the visualization attributes according to the name of each detector unit.

**Fig. 6** shows the display of the four BESIII sub-detectors in Unity. The corresponding display parameters of the materials have been updated to obtain a better display effect. The display attributes in Unity are set to be consistent with those in BesVis, but with more vivid and rich visualization effects that ROOT does not support. A display of the full BESIII detector in Unity is shown in Fig. 7.
Once the detector geometry and its visualization attributes are constructed in Unity, various applications can be further developed on top of it. The characteristics of Unity, including fantastic visual effects, multi-platform support and VR [42] or AR [43] devices integration, makes it convenient to develop event display tool, detector status monitoring software, and VR/AR applications for scientific research and education.

### V. FURTHER DEVELOPMENT OF APPLICATIONS

Once the detector geometry and its visualization attributes are constructed in Unity, various applications can be further developed on top of it. The characteristics of Unity, including fantastic visual effects, multi-platform support and VR [42] or AR [43] devices integration, makes it convenient to develop event display tool, detector status monitoring software, and VR/AR applications for scientific research and education.

#### A. Event display tool

Event display is a convenient tool for offline software tuning and physics analysis in HEP experiments. In Sect. II C, the application of Unity for event display in ATLAS and JUNO has been introduced. However, for both programs, the detectors were manually constructed in Unity independently, which took a lot of developers’ work.

With the method of automatic geometry transformation, the BESIII detector has been constructed in Unity. To develop the event display tool based on Unity, another work is to get the event data information, and to associate the fired detector units with their identifiers, which can be decoded from the name of each detector unit.

Once the association relationship is constructed, a set of scripts can be developed in Unity to control different visual effects for the fired and unfired detector units. Then the basic functions of event display tool have been realized. Fig. 8 shows a prototype of an event display tool under development in Unity and its rendering effects.

#### B. Detector and data monitoring

With the detector constructed in Unity, not only the offline event display but also the online monitoring software can be developed to help monitor the experiment’s operation status and the quality of the real-time data collected by the detector.

The detector units with abnormal operational status, such as dead or hot channels, can be distinctly displayed, which will help the shift operators to diagnose the potential detector problems.

Thanks to the multi-platform supports of Unity, a monitoring project can be easily deployed to different platforms and devices. Besides the traditional Windows and Linux operating system, the monitoring project can also be built into apps...
on the mobile platform, such as Android and iOS, so that the users can conveniently monitor the status of an experiment from their mobile phones or pads remotely.

C. Virtual reality applications

In roadmap for HEP software and computing R&D for the 2020s [2], application based on VR is an important direction of development [1]. VR simulates the user’s physical presence in a virtual environment. By presenting the emitting particles and their interactions with the detector, it can provide a new method with immersive experience for physicists to tune the offline simulation and reconstruction software, as well as performing physics analysis for rare events, as if the users are personally in the scene of the running detector. Some creative production has started in HEP community, such as ATLASrift [56] and BelleII VR [44].

With the method of automatic detector construction in Unity, and the extensible supports of VR devices from Unity, the VR applications for nuclear or HEP experiments can be easily developed. Since most nuclear or HEP experiments are not accessible during data taking because of safety or security requirements, our method has another advantage, allowing the general public to explore the detector with an immersive experience, watch the nuclear or HEP collisions in a realistic environment. It will be of great benefits to let the general public have a basic idea and understand what scientific research the nuclear physics and HEP community are doing.

D. Interdisciplinary applications

In addition to the applications in nuclear physics or HEP experiments, the method could also have wide applications in applied nuclear science, techniques, and industry.

For example, in nuclear power plant (NPP) monitoring, this method can be used to monitor the emitting neutrons or neutrinos, then to reconstruct the distribution of inner active fission area, so as to monitor the operation status of NPP.

In the field of nuclear medical imaging, the human body is more or less like a nuclear particle source or detector, but with dynamic geometry. Since our method of detector modeling is automatic, it can rapidly construct and visualize the human body, to allow observation of its status after interaction with nuclear particles in real time or semi real time. Other potential interdisciplinary applications include muon tomography [57–59], X-ray inspection, and fusion diagnosis, etc.

VI. SUMMARY

Detector description and visualization are essential techniques in software development for the next generation HEP experiments. Unity is a popular and versatile multi-media creation platform, which provides fancy visual effects, multiple platform supports and extensibility to VR and AR applications. For large-scale scientific projects, building the complicated detector with up to millions of components in Unity, while keeping them consistent with the geometry in offline data processing software, is very difficult and takes a lot of work for software developers.

A method of automatic detector data transformation from GDML to Unity is presented, which can construct the 3D modeling of detectors in Unity for further applications. The method has been realized in the BESIII detector with hundreds of thousands of components, and can be used in future experiments like CEPC. In addition to HEP experiments, the method also has great potentials in multiple interdisciplinary applications, education and outreach.

ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China (Grant No.11975021, 12175321, 11675275, U1932101), National Key Research and Development Program of China under Contracts Nos. 2020YFA0406300, 2020YFA0406400, the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDA10010900, the Chinese Academy of Sciences (CAS) Large-Scale Scientific Facility Program, the Fundamental Research Funds for the Central Universities, Sun Yat-sen University, the National College Students Science and Technology Innovation Project and the Undergraduate Base Scientific Research Project of Sun Yat-sen University.

REFERENCES

[1] M. Bellis et al. HEP Software Foundation Community White Paper Working Group — Visualization. 2018. DOI: 10.48550/ARXIV.1811.10309.

[2] J. Albrecht et al. “A Roadmap for HEP Software and Computing R&D for the 2020s”. Computing and Software for Big Science 3 (2019), p. 7. DOI: 10.1007/s41781-018-0018-8.

[3] The ATLAS Collaboration. “The ATLAS Experiment at the CERN Large Hadron Collider”. JINST 3.08 (2008), S08003–S08003. DOI: 10.1088/1748-0221/3/08/s08003.

[4] The CMS Collaboration. “The CMS experiment at the CERN LHC”. JINST 3.08 (2008), S08004–S08004. DOI: 10.1088/1748-0221/3/08/s08004.

[5] L. Evans. “The Large Hadron Collider”. New Journal of Physics 9.9 (2007), pp. 335–355. DOI: 10.1088/1367-2630/9/9/335.

[6] R. Chytracek, J. Mccormick, W. Pokorski, et al. “Geometry Description Markup Language for Physics Simulation and Analysis Applications”. IEEE Transactions on Nuclear Science 53.5 (2006), pp. 2892–2896. DOI: 10.1109/TNS.2006.881062.
