EPJ Web of Conferences **245**, 08003 (2020) https://doi.org/10.1051/epjconf/202024508003
CHEP 2019

Belle2VR: An Interactive Virtual Reality Visualization of GEANT4 Event Histories

Leo Piilonen[1,*], *Zachary* Duer[1], and *George* Glasson[1]

[1]Virginia Polytechnic Institute and State University, Blacksburg VA 24061, USA

Abstract. Belle2VR is a novel interactive virtual reality visualization of the Belle II detector at KEK and the animation therein of GEANT4-simulated event histories. The user, wearing a VR headset, manipulates a gamepad or hand controller(s) to interact with and interrogate the detailed GEANT4 event history over time, to adjust the visibility and transparency of the particles and detector subsystems, to translate freely in 3D, to zoom in or out, and to control the event-history timeline. In this way, the user explores the world of subatomic physics via electron-positron collision events in the Belle II experiment at the SuperKEKB colliding-beam facility at KEK in Japan. Developed at Virginia Tech by an interdisciplinary team of researchers in physics, education, and virtual environments, the simulation is intended to be integrated into the undergraduate physics curriculum.

1 Introduction

The Belle II experiment [1] at the High Energy Accelerator Research Organization [3] (KEK) in Tsukuba, Japan studies the properties of heavy quarks and leptons and searches for evidence of new phenomena [2] beyond the Standard Model [4] of elementary particle physics by recording individual electron-positron collisions produced by the SuperKEKB [5] colliding-beam accelerator. The Belle II Analysis Framework 2 (`basf2` [6]) is used to reconstruct and analyze these collision events. This framework incorporates the ability to generate synthetic events and then simulate, via the `GEANT4` [7] toolkit, the Belle II detector's response to the daughters of the parent collision. By combining all of these steps, a basf2 user can generate, simulate, reconstruct and analyze a reference dataset for detailed aggregate comparisons with the billions of recorded events. The `Belle2VR` app [8], developed with the Unity game engine [9] by an interdisciplinary team supported by the Institute for Creativity, Arts and Technology [10] at Virginia Tech, imports the detailed `GEANT4` history of each simulated event for visualization in virtual reality (VR), allowing users to explore this history as well as the interactions of the daughter particles in each part of Belle II detector.

2 Detector Geometry

The `GEANT4` package provides a comprehensive set of tools to simulate accurately the passage of elementary particles through matter. It incorporates a geometry- and material-description

*e-mail: piilonen@vt.edu

system that is used by the basf2 developers to construct a detailed model of the Belle II detector, including all active detector components (i.e., those that provide a recordable response in the real detector to the passage or interaction of a particle) and all passive structural elements.

The GEANT4 geometry consists of three-dimensional volumes, each defined by a shape (selected from a predefined set—boxes, cylinders, etc.) and material (air, steel, etc.) and placed within an enclosing volume. In a complete geometry definition, the "world" parent of all volumes is typically a large air-filled rectangular parallelepiped—a box—centred at the origin and aligned along the axes of the Cartesian coordinate system. The nesting of volumes may be shallow or deep, i.e., one to many levels below the world volume. In a proper geometry definition, a daughter volume is contained entirely within its parent volume. The designer specifies the dimensions, placement and orientation of each volume and then fills it with a suitable material (selected from a predefined set or constructed programmatically from elements, compounds or mixtures). Some volumes are declared *sensitive* so that their responses to the passage of a particle are recorded during the GEANT4 simulation. A typical geometry is defined once and then never modified in its subsequent usage for particle-detector interactions, since there is a substantial time penalty associated with any redefinition or modification of this geometry.

The GEANT4 geometry description is encoded in a manner that is not recognized by any other 3D visualization toolkit or application programming interface (API) like OpenGL. [11] Instead of native three-dimensional shapes like boxes, cylinders, or cone frustra, these toolkits define a 3D shape solely by its surface(s); each surface, in turn, is defined by its tessellation into a complete void-free set of non-overlapping triangles. Each triangle is specified by its three vertices and is guaranteed to be flat in 3D, a prime requirement for high-performance manipulation and rendering of the surface.

Fortunately, GEANT4 provides the method GetPolyhedron to render each of its volumes as its triangle-tessellated surface(s). We wrote two basf2 modules [12] to export the complete set of polyhedra, preserving the original hierarchical structure, in the Virtual Reality Markup Language [13] (VRML2) and Filmbox [14] (FBX®) formats. *Non-sensitive* volumes that were declared invisible or contain gaseous material are not exported by these modules. We do not use the available animation features of these formats since the GEANT4 geometry is static. The exported plain-text files can be displayed in any compatible 3D viewer app (e.g., FBX Review [15] for multiple platforms, Cheetah3D [16] for Mac OS X, or LynX 3D [17] for Windows). The FBX files are imported directly into the Unity development platform to define the Belle II detector geometry in the Belle2VR app.

2.1 Event Histories

GEANT4 provides the user hook G4UserSteppingAction to carry out custom actions at each step of each particle in the simulation of a complete event. The basf2 framework engages this hook in its SteppingAction method. We add some code to SteppingAction to write the per-step information to a VR-output file, and some code to the basf2 start-of-event and end-of-event methods in EventAction to open/close a new VR-output file for each simulated event.

Most, but not all, steps are written to the VR-output file. No steps are written if (a) the particle is a nucleus heavier than helium-4, or (b) the simulation time exceeds 100 ns, or (c) the kinetic energy is below 500 eV (not applicable to *optical photons* whose energy is zero).

We record the following G4Step information: the track identifier; the parent track's identifier; the name, mass and charge of the particle; the current step number; the GEANT4 status,

process type (e.g., "Transportation" or "Decay"), energy deposition, and first-step and last-step flags for the particle in this step; the volume through which the particle is moving (and a Belle II-specific identifier for a *sensitive* volume); the volume's material; and the position, time, momentum and energy of the particle at the start and end of the step. This information is written in one `csv`-compatible line to the VR-output file. A typical $e^+e^- \to \mu^+\mu^-$ event contains about 15,000 lines while a typical $e^+e^- \to B\bar{B}$ event contains about 60,000 lines.

Each VR-output file is post-processed by a `perl` script to prepend records for the SuperKEKB accelerator's electron and positron beam-line bunches (represented by single electrons and positrons), and then sort all of the records by the track identifier, the step's start-time, and the step's end-time. This sort order is chosen to minimize the amount of per-frame calculation time needed by Belle2VR to animate the event. This file can be examined in any spreadsheet or text editor. Optionally, the file can be `gzip`-compressed to reduce its size by a factor of roughly 10.

Belle II users with access to the `basf2` source code can create their own VR-output files by applying the aforementioned customizations to the simulation module's `SteppingAction` and `EventAction` methods and then using the `basf2` event-generation and -simulation prescriptions to generate and simulate the desired standard or custom physics processes.

The VR-output files are stored in the `events` folder at the same level as the Belle2VR app. The plain-text steering file `events.lis` in the same folder specifies which VR-output files are to be animated and in which order. In this steering file, each animation event is specified by (a) the label to be shown in the in-app *Particles* menu, (b) the filename (without the `.gz` suffix, if compressed), and (c) the filename of a PNG-format image file corresponding to the event process (e.g., $e^+e^- \to \mu^+\mu^-$) and displayed in the app during the event's animation. If this folder contains the plain-text file `events.url` and this file points to a URL that contains the above contents (VR-output files and a steering `events.lis` file), then the animated events are read from this URL rather than the local folder.

As a fall-back, if the above mechanisms fail (e.g., if the `events` folder does not exist), the app animates five embedded events.

2.2 Event Animation

A snapshot of the animation of a typical event is shown in Fig. 1, just after the primary e^+e^- collision.

Before starting the animation of one simulated event, every step in the event history is drawn as a faint (straight) line, color-coded according to the particle type. These faint lines persist throughout the animation. Then, the animation commences at 10 ns before the primary electron-positron collision and proceeds until 100 ns after this primary collision. (Beam-line bunch crossings occur at 4-ns intervals; most crossings do not result in a collision.) Time is stretched so that this animation takes about one minute of user time.

Each particle in the animation is shown as a color-coded sprite with the particle-type label. Anti-particles are white with a color-coded border (rather than black). Parton sprites are circles; meson (baryon) sprites are two (three) overlapping circles. Charged (neutral) particles have a solid (dashed) border. A particle's sprite is shown in the animation only after its creation time in the `GEANT4` simulation; it is dimmed (or invisible, if desired by the user) after its final step. An animated trail is displayed behind each sprite to guide the user's eye to its motion.

A small cube-shaped hollow sprite is drawn to indicate the passage of a particle through a *sensitive* volume in the Belle II detector with the deposition of some energy therein. These "hits" are initially dimmed. They are brightened in the animation when the particle enters the *sensitive* volume, and a detector-specific sound is emitted at this time. The Belle2VR

user can learn to correlate these distinct sounds with the sensitive subsystems (e.g., ECL or electromagnetic calorimeter) of the Belle II detector. After 100 ns of animation have elapsed, Belle2VR removes the faint lines, sprites, and motion trails and displays only the detector hits for a few seconds. In this way, the user can see the limited information that is recorded by Belle II in the snapshot of an electron-positron collision event.

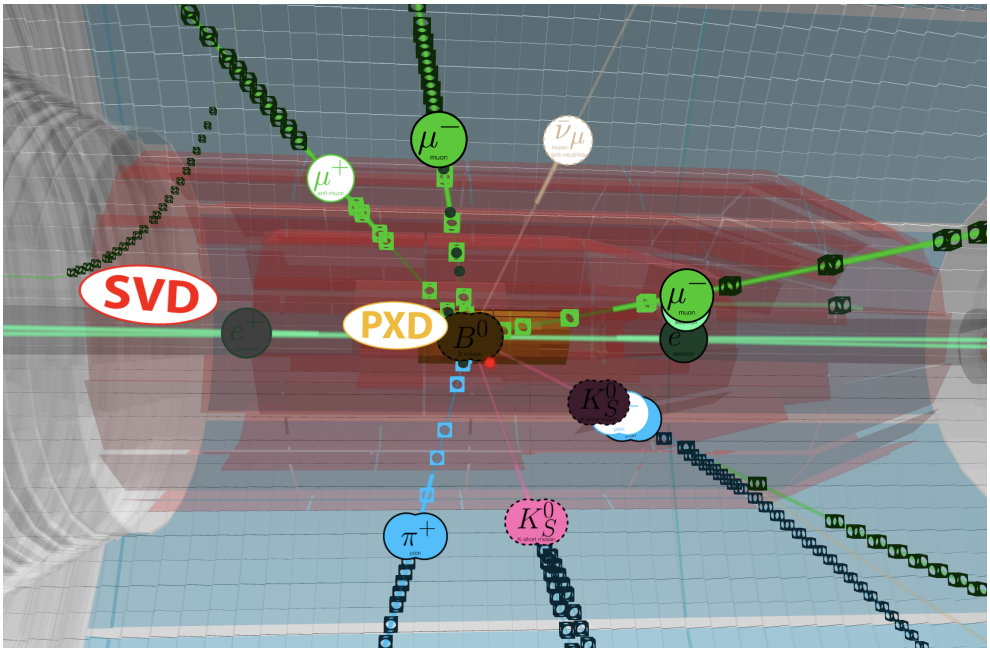

Figure 1. Snapshot of the animation of a typical $e^+e^- \to \Upsilon(4S) \to B^0\bar{B}^0$ event, where $B^0 \to J/\psi\, K^0_S \to (\mu^+\mu^-)\,(\pi^+\pi^-)$ and $\bar{B}^0 \to D^+\mu^-\nu_\mu$. The snapshot shows the faint lines for the entire event history, the sprites for and trails behind each particle, and the hollow cubes at each of the hits in the sensitive elements of the detectors. The B^0 sprite is at the centre of the pixel detector (PXD) that is contained within the five-layer silicon vertex detector (SVD).

2.3 User Operation

The user dons a VR headset (Oculus Rift or HTC Vive, connected to a Windows PC or standalone Oculus GO) and starts the app. After a brief animation of the SuperKEKB accelerator while the app loads, the view switches to the full-scale Belle II detector encased in the iron yoke that confines the magnetic field of the Belle II solenoid. The detector is positioned within a four-storey room called the Cube at the Virginia Tech Moss Arts Center since Belle2VR was originally developed for simultaneous use by multiple headset-wearing students moving about and interacting with each other in this room while inspecting the Belle II detector and event animations. (Of course, the proper setting for Belle II would be the Tsukuba Experimental Hall at KEK.) The user holds a gamepad (Xbox-compatible) or the headset-specific hand controllers to control the animation and interrogate the dynamic features of the displayed particles.

In the Cube, users wear an MSI VR One backpack laptop and an Oculus Rift VR headset with an attached constellation of motion-capture markers. Each VR backpack runs an independent instance of the application. A 24-camera Qualisys optical motion-capture system

tracks the motion-capture markers, recognizing each user's constellation of motion-capture markers as a unique object. The Qualisys Track Manager software transmits over WiFi on a local area network the 3D position and rotation of each recognized object to each running Belle2VR application. The position of each user's perspective, represented as a virtual camera in Unity, is controlled by the received motion-capture coordinates for the respective object that represents their headset and motion-capture marker constellation. The rotation of the user's perspective is controlled by the inertial measurement unit, comprised of accelerometers, gyroscopes, magnetometers, and other sensors within each head-mounted display unit.

When operated by an individual outside the Cube, the vendor-provided integrated trackers determine the headset's position and orientation (and, for the Vive, the hand-controllers' position and orientation) to control the user's position and orientation within the VR world.

In the absence of a VR headset, the user can run the native application on a computer (Windows, Mac OS X, or Linux) or the WebGL app within a web-browser window and control the animation using the keyboard.

A targeting reticle—a green dot in the middle of the field of view—is used to interact with the dynamic elements of the VR world. As the user reorients, the reticle moves, landing on the closest visible object. If this object is interactive (e.g., a particle sprite), it is highlighted to indicate that it can be interrogated or manipulated.

When the user interacts in this way with a particle sprite, a panel appears above it to display useful information about the particle: its type, its track identifier, and its energy and momentum (initial, current, and final). This panel moves with the particle until it is dismissed. If the user targets and selects the *Focus* button on this panel, only this particle's direct ancestors and children and displayed until the user targets and selects the now-relabelled *Unfocus* button. If the user targets and selects the *Save* button on this panel, the above information is saved to one row of a large virtual panel on one wall of the enclosing room. The user can then interact with these virtual-panel rows, selecting some of them for summation; in this way, the user can compare the values for a parent particle with the sum values for its daughters and thereby verify conservation laws for electric charge, energy, and momentum.

A two-dimensional diegetic menu within the VR world can be summoned by pressing a gamepad button to manipulate many other settings. This floating menu has three sub-menus, labelled *Transform*, *Detector*, and *Particles*. After the manipulations, the user dismisses the menu. At the bottom of all of these menus, the animation timeline is displayed with coarse (power-of-ten) and fine scales for the pace of the animation. Using the reticle, the user can scrub forward and backward in the timeline, adjust the animation pace, pause the animation, or to repeatedly loop the animation of a single event (with chosen start and end animation times to override the default values of -10 ns and +100 ns).

The *Transform* menu permits the user to scale the detector's size, to move the detector within the enclosing room, or to reset these transformations to their default settings.

The *Detector* menu permits the user to adjust the transparency and visibility of each of the Belle II detector elements, with one sub-panel for the "Active" (i.e., *sensitive*) elements and another for the "Passive" (i.e., structural) elements.

The *Particles* menu lets the user control the size of the particle sprites, to select which particle types in the current event should be visible or hidden, to show or hide the particles that are dead (i.e., whose histories are entirely in the past relative to the current animation time), and to show or hide the detector hits. This menu also lets the user choose which event to animate from the catalog of events.

2.4 Conclusion

The features of the Belle2VR app, its development process, and its pedagogical aspects, are described in Refs. [18] and [19]. Up-to-date information about the current state and availability of the free Belle2VR app is available at Ref. [20]. Belle2VR has been featured and well received in public interactive displays at about 30 institutions. We continue to develop and add features to the app to better integrate it into the undergraduate particle-physics curriculum at Virginia Tech, to better enable its use in MasterClasses [21] for particle-physics novices, and to make the app even more appealing to the general public when used in open-house and similar events [22] at national laboratories and universities.

References

[1] T. Abe *et al.* (Belle II Collaboration), *Belle II Technical Design Report*, (2010). arXiv:1011.0352

[2] E. Kou and P. Urquijo, eds. *et al.* (Belle II Collaboration), *The Belle II Physics Book*, Prog. Theor. Exp. Phys. **2019**, 123C01 (2019). doi:10.1093/ptep/ptz106, arXiv:1808.10567

[3] kek.jp/en/

[4] en.wikipedia.org/wiki/Standard_Model

[5] K. Akai, K. Furukawa, and H. Koiso, *SuperKEKB Collider*, Nucl. Instrum. Meth. in Phys. Res. A **907**, 188 (2018). doi: 10.1016/j.nima.2018.08.017, arXiv:1809.01958

[6] T. Kuhr, C. Pulvermacher, M. Ritter, T. Hauth and N. Braun (Belle II Framework Software Group), *The Belle II Core Software*, Comput. Softw. Big Sci. **3**, 1 (2019). doi: 10.1007/s41781-018-0017-9, arXiv: 1809.04299

[7] S. Agostinelli *et al.*, *GEANT4—A Simulation Toolkit*, Nucl. Instrum. Meth. in Phys. Res. A **506**, 250 (2003). doi:10.1016/S0168-9002(03)01368-8, geant4.web.cern.ch

[8] Z. Duer, L. Piilonen and G. Glasson, *Belle2VR: A Virtual-Reality Visualization of Subatomic Particle Physics in the Belle II Experiment*, IEEE. Comp. Graph. and Appl. **38**, 33 (2018). doi: 10.1109/MCG.2018.032421652

[9] unity3d.com

[10] icat.vt.edu

[11] www.opengl.org

[12] The exporter source code is available at github.com/HSF/Visualization

[13] en.wikipedia.org/wiki/VRML

[14] www.autodesk.com/products/fbx/overview

[15] www.autodesk.com/products/fbx/fbx-review

[16] cheetah3d.com

[17] ozone3d.net

[18] IEEE Comp. Graphics and Appl. **38(3)** (2018) 33.

[19] vimeo.com/220004044

[20] www.phys.vt.edu/~piilonen/VR/

[21] physicsmasterclasses.org

[22] event.gg/9494/ (2018), aip.org.au/event/universal-science/ (2019)