art Visualization Using ParaView

Keshav Kapoor (Naperville Central High School)
Dr. Adam Lyon (Fermi National Accelerator Laboratory)

Abstract

The purpose of this project is to create a real time visualization of the data that is produced from the *art* event framework created by the Fermilab Scientific Computing division. to check the geometry of the experiment as well as preparing *art* to be used with High Performance Computing (HPC). To accomplish this task we will be using a visualization tool called ParaView which was created by Kitware, Los Alamos National Lab, and Sandia National Lab. We already have a pipeline between the *art* framework and paraview which allows us to display simple objects as they are created. This pipeline was written in python and I will be converting it into C++ as to keep it constant with the rest of the *art* framework. This will hopefully give experiments a tool they can use to check their reconstruction algorithms and detector designs.

Introduction

Background

The fields in which this project takes place are High Energy Physics (HEP) and Supercomputing visualization software. With High Energy Physics, we are specifically using *art*. This is an event processing framework written in c++ which was created by Fermilab's Scientific Computing Division (Kutschke). In *art* there is an event loop, where a single event goes through the loop and is modified and analyzed at different modules in the loop. There are five types of modules: analyzer, producer, filter, source, and output (Kutschke). The analyzer inspects information in the event, the producer can inspect and add information to the event, the filter can tell *art* to skip modules for the event, the source reads events from a source, the output takes selected data products and outputs them (Kutschke). Paraview is a supercomputing visualization software that we will be using to visualize the event data. Paraview uses the Visualization Toolkit, a C++ toolkit to help create 3D visualizations, to create its visualizations(Ayachit). Paraview has a

graphical user interface for the user to manipulate the object created from the data line changing color, viewing angles, and specularity. Paraview also has a feature called catalyst which does In Situ visualization, where paraview does the visualization in the system and then outputs images, instead of outputting data, this is due to the bottleneck created by the i/o, making the system faster and more efficient if there were less reads/writes(Bauer, Geveci, Schroeder).

Context of this work

This project will allow *art* to come with a 3D visualization tool, which can then be used to check geometry on and display events. Previously, the visualization of the events were done by individual scientists based on their own experiments, with this new addition scientists will not have to create their own visualization tools. Over the summer we will be focusing on getting a pipeline for the data between *art* and the visualization software Paraview while also keeping the an *art* -like structure. This separation will also allow for a filtering of the data before using it to make a grid. Later this software can be used by experiments to check their detector and reconstruction algorithms.

Methods

General Outline of Program

My tasks included making the code to be more inline with the *art* framework. The visualization data used to all be gotten and passed in an analyzer module. In *art*, data between modules cannot be accessed with get functions. The data has to be put in something called a data product, which is a class, that is then put inside the event. The data product holds data produced and analyzed as the event passes through the modules. It generally only holds simple data types like integers, doubles, or floats. This data product can then be accessed by other modules to use the data inside it.

In a producer module, I first got data from each step in the tracks and put them in simple data vectors, like int, double, and float vectors. These vectors are put into the data product which I created. Later on in the event loop, an analyzer accesses the data product. After it is able to access the data, the analyzer makes arrays of VTK data types with the data gotten from the data product. It then takes the VTK data and puts them in grid which can be viewed in ParaView, and passes the grid onto the Catalyst pipeline so that ParaView can get the data.

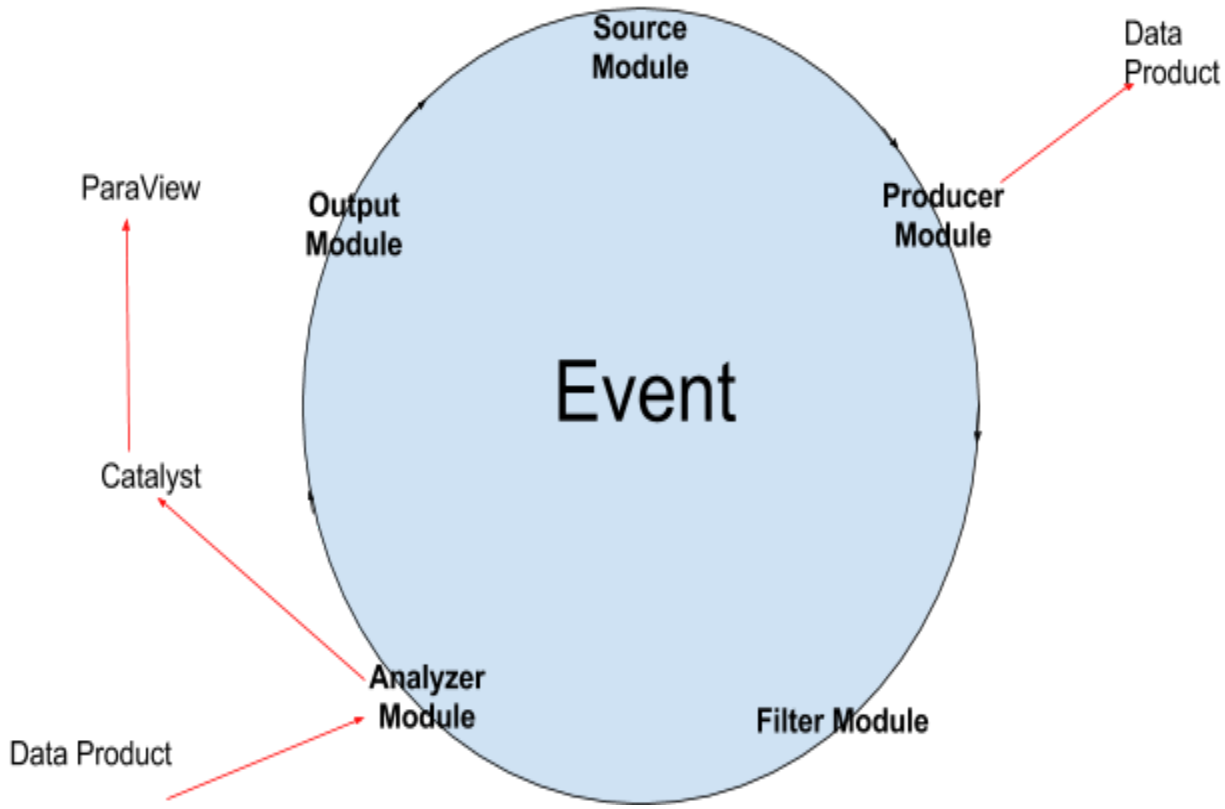
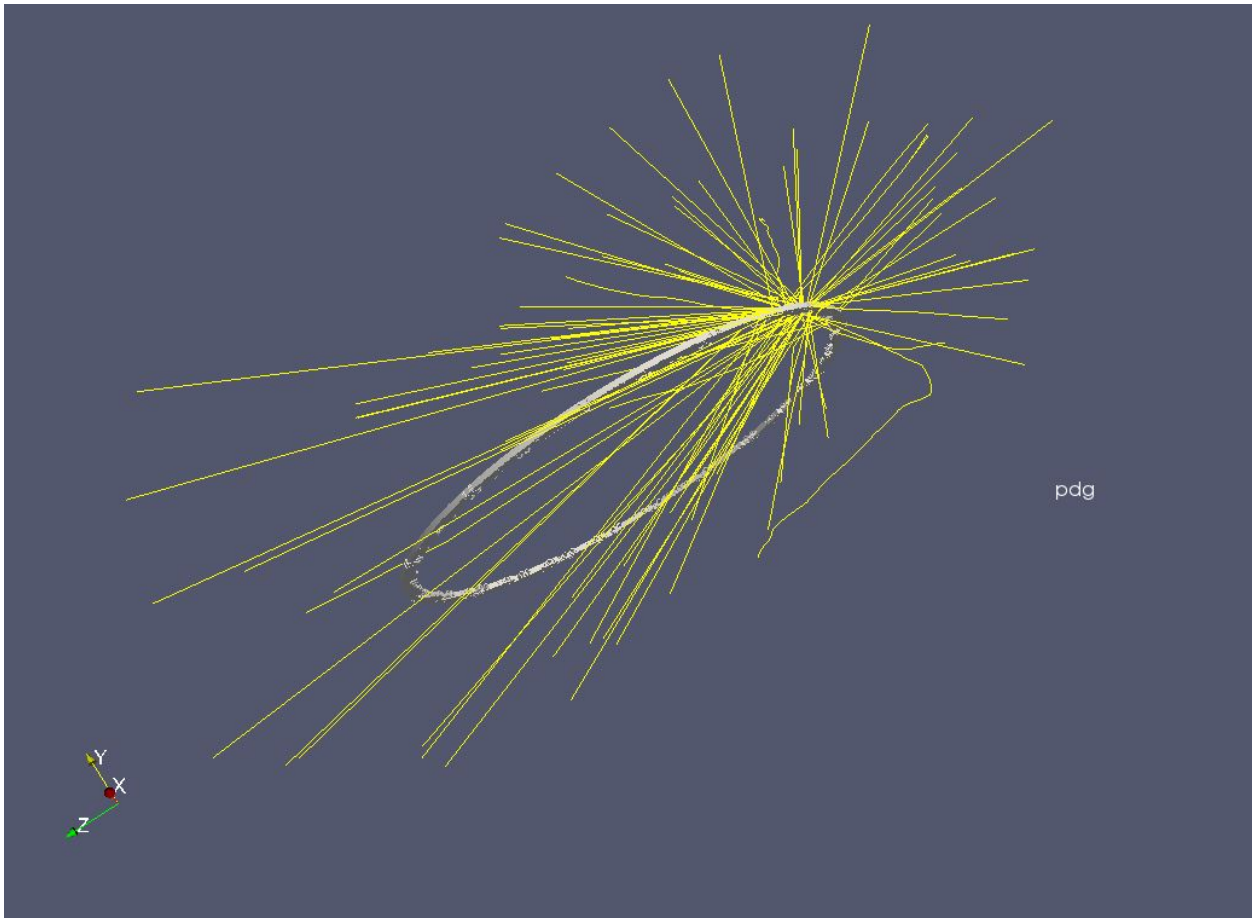
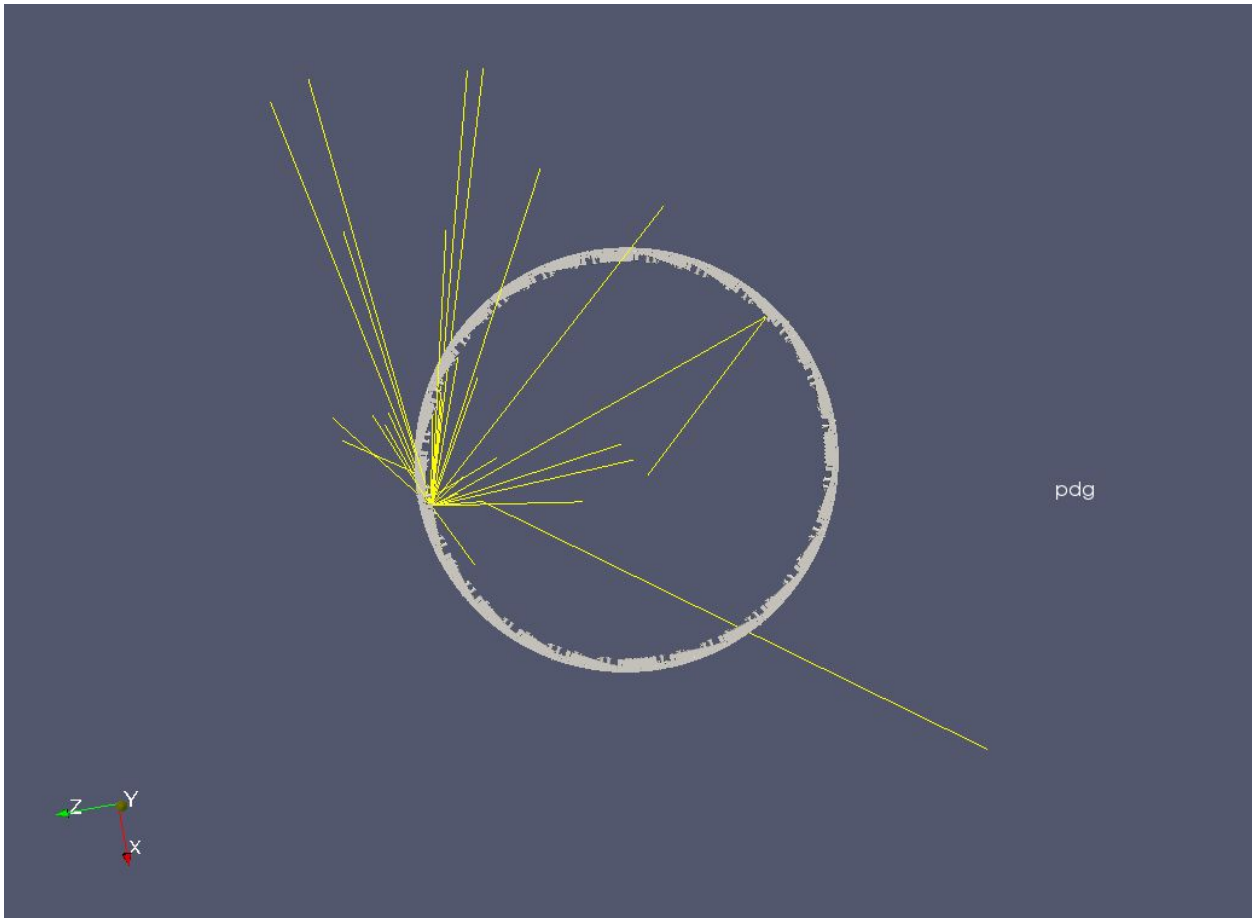


Figure 1: A visual representation of the event loop showing how the data is transported through the event loop. The red arrows represent the flow of the data. The data product is a part of the event that gets passed on to every module.

Pictures





Future

There are many more aspects of the program which should be improved. The first item we want to improve is the pipeline between *art* and ParaView. This pipeline currently is a service written in python which a module can call. We want to make this service a part of an analyzer module and write it in C++. In the near future, the experiment protoDUNE is looking into using this software to visualize their simulations.

Acknowledgements

Fermilab is operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy. QuarkNet is an educational program sponsored by the National Science Foundation and the Department of Energy whose aim is to support science education in schools by establishing a nationwide

network of science teachers. This work was supported by CRADA FRA-2014-0022-02 agreement between Fermi Research Alliance LLC and the University of Notre Dame for the 2015-16 QuarkNet summer program.

I would also like to thank Dr. Adam Lyon and Paul Russo for their help and guidance on this project. I would not have been able to get through the C++ without Paul.

References

1. Ayachit, U. (2016, June 17). *The ParaView Guide*.
2. Bauer, A. C., Geveci, B., & Schroeder, W. (2015, February). *The Catalyst User's Guide v2.0 ParaView 4.3.1* [User's guide to Catalyst].
3. Kutschke, R. (2016, July 18). *Intensity Frontier Common Offline Documentation: Art Workbook and Users Guide*.