Universidad Michoacana de San Nicolás de Hidalgo.
Facultad de Ciencias Fisico Matemáticas
"Mat. Manuel Rivera Gutierrez"

Diseño, desarrollo e implementación de un seleccionador y visualizador de eventos (Event Display/Browser) para la colaboración internacional del experimento HAWC

TESIS

Que para obtener el grado de
Maestro en Ciencias en Ingeniería Física

Presenta
Cederik León De León Acuña

Asesor
Dr. Umberto Cotti Gollini.

Morelia, Michoacán, Agosto de 2013.

Resumen

Se presenta el diseño y desarrollo de software para seleccionar y visualizar los eventos detectados por el observatorio de rayos gamma HAWC (High Altitude Water Cerenkov) mediante la carga de archivos digitales en formato ROOT.

El software permite al usuario trabajar con una selección arbitraria de eventos detectados por el observatorio o contenidos en librerías de datos simulados; facilita la manipulación de los mismos y la obtención de la información en ellos contenida.

Agradecimientos

Agradezco a toda la comunidad nicolaita

ÍNDICE GENERAL

ÍNDICE DE CUADROS

ÍNDICE DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

1.1. sedeb

En el presente trabajo se muestra el proceso de diseño, desarrollo e implementación de una herramienta computacional que permite acceder a la información relacionada a los eventos detectados por el observatorio HAWC, condicionar la búsqueda de eventos bajo criterios y necesidades específicas, lo anterior en un entorno gráfico.

La datos accedidos pueden ser desplegados y generar un conjunto de gráficos con información relevante para el usuario (histogramas, mapas de eventos sobre el detector, gráficas X-Y)

1.2. Proyecto HAWC

El Observatorio HAWC [1],[2] (High-Altitud Water Cherenkov) es una instalación diseñada para observar los chubascos atmosfericos producidos por rayos cósmicos y rayos gamma con energía del orden de TeV (10^{12} eV), con una apertura instantánea que cubre más del 15 % del cielo. Con este amplio campo de visión, el detector está expuesto a la mitad del cielo durante un período de 24 horas.

Es una colaboración internacional entre instituciones de los Estados Unidos y México, la Universidad Michoacana de San Nicolás de Hidalgo entre ellas[3].

El Observatorio actualmente genera informacion que almacenada, en archivos de

datos, es del orden de Terabytes; de aquí la importancia de la implementación de software que permita seleccionar eventos, almacenados en esos archivos, aplicando criterios definidos por el usuario. El desarrollo de un visualizador de eventos soluciona el problema de selección y despliegue de datos, lo cual acelera el proceso de análisis.

Los rayos gamma del orden de TeV son marcas de los entornos más extremos del universo conocido: explosiones de supernovas, núcleos de galaxias activas y erupciones de rayos gamma.

Se piensa que los rayos gamma están correlacionados con los lugares donde son acelerados los rayos cósmicos cargados, cuyo origen ha sido un misterio durante cerca de 100 años.

HAWC actualmente se encuentra en construcción, en los costados del volcán de la Sierra Negra cercano a la ciudad de Puebla, México. Ubicado en una altitud de 4100 metros, HAWC será utilizado para realizar estudios sinópticos del cielo en longitudes de onda entre los 100 GeV y los 100 TeV.

Los datos adquiridos por el observatorio HAWC son puestos a disposición de los científicos que participan en la realización del experimento, con el fin de que sean analizados; la herramienta de software que permite realizar el análisis y reconstrucción de eventos está contenida en un conjunto de programas llamado aerie (Analysis and Event Reconstruction Integrated Environment).

CAPÍTULO 2

FÍSICA DE PARTÍCULAS Y ASTROPARTÍCULAS

2.1. Modelo Estándar de las partículas fundamentales

El Modelo Estándar incorpora todo lo conocido sobre el mundo subatómico, identifica los constituyentes elementales de la materia, por lo tanto también describe la manera en como interactúan entre ellos y como se transforman.

Los primeros pasos en el desarrollo del Modelo Estándar fueron dados durante los años 1960, cuando Sheldon Glashow introdujo una manera de unificar las interacciones electromagnética y débil; en 1967 Steven Weinberg y Abdus Salam incorporan el mecanismo de *Higgs* a esta teoría. Para 1973 los experimentos en aceleradores de partículas confirmaron que los elementos que conforman a los *Hadrones* son los *Quarks*. También el *Modelo Estándar* predijo (posteriormente confirmado) la existencia y masa de los Bosones Z y W[4].

El Modelo Estándar de las partículas fundamentales es una teoría ampliamente aceptada que ha predicho, con enorme precisión, la existencia y la fenomenología de las partículas existentes en el universo, describe cada tipo de partícula en términos de un campo matemático, valiéndose de la teoría cuántica del campo.

2.1.1. Constituyentes del Modelo Estándar

Los constituyentes fundamentales del Modelo Estándar o conjunto de materia elemental son los fermiones, lo cuales son seis quarks y seis leptones de spin 1/2, adicionalmente cada partícula tiene asociada su correspondiente anti-partícula, la cual tiene las mismas propiedades pero con cargas opuestas. Los quarks a diferencia de los leptones, tienen además de la carga eléctrica, la carga de "color". Los neutrinos son leptones que no poseen carga eléctrica.

Quark	**AntiQuark**	**Leptón**	**Anti Leptón**
u, (up)	$\bar{\mathbf{u}}$, (antiUp)	ν_e , (neutrino electrón)	$\bar{\nu_e}$, (Antineutrino electrón)
d, (down)	$\bar{\mathbf{d}}$, (antiDown)	e^-, (electrón)	e^+, (positrón)
c, (charm)	$\bar{\mathbf{c}}$, (antiCharm)	ν_μ, (Neutrino del muón)	$\bar{\nu_\mu}$, (AntiNeutrino muón)
s, (strange)	$\bar{\mathbf{s}}$, (antiStrange)	μ^-, (muón)	μ^+, (Antimuón)
t, (top)	$\bar{\mathbf{t}}$, (antiTop)	ν_τ,(Neutrino tau)	$\bar{\nu_\tau}$, (AntiNeutrino tau)
b, (bottom)	$\bar{\mathbf{b}}$, (antiBottom)	τ^-, (tau)	τ^+, (AntiTau)

Cuadro 2.1: Fermiones, constituyentes del Modelo Estándar

Según el *Modelo Estándar*, la manera en que interactúan estas partículas, entre sí, está mediada por los *Bosones* de norma, que son particulas que no obedecen el "principio de exclusión de Pauli" es decir, obedecen la estadística de *Bose-Einstein*, donde varias partícuas pueden compartir un mismo estado cuántico. Estos Bosones de norma, llamados también partículas "mediadoras" son los que permiten la interacción y generan los procesos que se dan lugar en el *Modelo Estándar*; según el tipo de fuerza fundamental que rige ciertos procesos se le asocia su correspondiente Bosón:

[5]

2.2. Rayos gamma

Los fotones son partículas fundamentales del modelo estándar, pertenecen a la familia de los bosones, (partículas mediadoras de alguna de las fuerzas fundamentales de la física), en este caso se encargan de mediar entre las interacciones en las que

Boson	**Fuerza**
$W^{\pm}$, Z^0	Débil
γ, (Fotón)	Electromagética
g, (Gluón)	Fuerte
G, (Gravitón)	Gravitacional(no detectada)

Cuadro 2.2: Bosones, mediadores en el Modelo Estándar

interviene la fuerza electromagnética; no poseen carga eléctrica ni masa.

A los fotones con energías más allá de los 100 keV se les llama rayos gammas de alta energía.

2.2.1. Producción de rayos gamma

Uno de los procesos que más contribuye a la producción de rayos gamma que el observatorio detecta, es mediante la interacción protón-protón, con la subsecuente producción de un meson pi neutro, y el consecuente decaimiento de mesón en dos fotones que se llevan consigo gran parte de la energía del proceso; para ello los protones deben de ser acelerados mediante mecanismos como el de sincrotron ó el de Fermi de 2o orden.

2.2.2. Astronomía de rayos gamma

Los rayos gamma, al igual que los rayos cósmicos, nos permite observar objetos astrofísicos distantes, estas particulas poseen información sobre las fuentes que las emiten; en el caso de los gamma, estas son producidas por interacciones hadrónicas y electromagnéticas. Como fuentes de rayos gamma se encuentran las novas, supernovas, remanentes de estrellas, pulsares.

La astronomía de baja energía emplea la siguiente sección del espectro electromagnético de para realizar sus observaciones y se muestra en la siguiente tabla:

Esta incluye la radiosastronomía, la astronomía optica y la de rayos X.

Por encima de los 100 keV los rayos gamma se consideran de alta energía, y con su detección se da origen a la astronomía de rayos gamma.

Tipo de radiación	**Rango de energía**
Ondas de radio	10 micro eV hasta 1 mili eV
Infrarrojo	1 mili eV hasta los 1 eV
Luz visible	1 eV hasta los 10 eV
Ultravioleta	10 eV hasta los 100 eV
Rayos X	100 eV hasta los 10 keV
Rayos gamma de baja energía	hasta 100 keV

Cuadro 2.3: Rangos de energía para astronomía de baja energía

Los experimentos que lograron detectar gammas con energías comprendidas entre los 100 MeV y 10 GeV son SAS-2 (Small Astronomy Satellite) en 1972 y hasta 1990 con EGRET (Energetic Gamma Ray Experiment Telescope)[6].

2.2.3. La física de la detección de rayos gamma

Cuando los rayos gamma se encuentran en su camino con la tierra, estos interactuan con ella atraves de la atmósfera,

Al no poseer ni masa ni carga, los gamma pueden recorrer grandes distancias sin ser afectados por campos eléctricos o magnéticos, esto permite tener una mayor certeza sobre su origen o fuente emisora, sin embargo los gammas pueden sufrir absorción al interactuar con la materia.

Aparte del método de detección directa de estos rayps gamma por medio de satélites atmosféricos, en los demás métodos efectos intervienen efectos físicos producidos por la interaccion de los rayos gamma con la atmósfera. La física involucrada separa los experimentos en dos grandes categorías y ambas están ligadas a la produccion del chubasco atmosférico inducido por el rayo gamma como partícula primaria. Unos observatorios se caracterizan para detectar el frente del chubasco atmosférico (HAWC, AUGER, TIBET por ejemplo), otros se caracterizan por detectar los fotones emitidos por la radiación Cherenkov producida a lo largo del desarrollo de del chubasco atmosférico (HESS, MAGIC, VERITAS)

Experimento	**Rango de energía**	**Tipo de experimento**	**Ubicación**
VERITAS	50 GeV - 50 TeV	Telescopios Cherenkov Atmosféricos	Arizona, USA
TIBET AS	TeV - 100 PeV	Centelleo	Tibet, China
HESS	100 GeV - 10 TeV	Telescopios Cherenkov Atmosféricos	Namibia
HAWC	100 GeV 100 TeV	Tanques Cherenkov de Agua	Sierra Negra, México

Cuadro 2.4: Rangos de energía de algunos experimentos

CAPÍTULO 3

SEDEB: SIMPLE EVENT DISPLAY EVENT BROWSER

3.1. Generalidades

sedeb : simple event display event browser (Simple Visualizador y Navegador de Eventos) es una herramienta computacional que permite al usuario visualizar la información de una colección de eventos generados por el observatorio HAWC, almacenada en uno o varios archivos en formato ROOT [7], esto de manera sintética, ordenada e intuitiva, mediante una interfaz gráfica (GUI).

sedeb muestra al usuario los eventos que han sido detectados por el observatorio HAWC y que han sido reconstruidos por las herramientas de aerie; graficamente señala los tanques y los tubos fotomultiplicadores (pmt) que han sido activados en el evento que se ha indicado y las variables referidas en la estructura de datos del archivo Root.

Es capaz de generar histogramas y gráficos a partir de la colección de variables que conforman la información del evento seleccionado; permite establecer cortes sobre las variables, esto significa que permite generar selecciones de datos particulares que poseen características impuestas por el usuario, desplegando en una lista solo aquellos eventos que satisfagan los criterios impuestos, esta lista permite acceder rapidamente a los eventos deseados.

Esta herramienta permite realizar estadística de los eventos contenidos en los archivos cargados en el programa sobre las variables contenidas en él; si se han aplicado cortes, la estadística corre sobre los eventos que satifacen los criterios impuestos

en los cortes.

Permite, mediante la herramienta de conversión de formatos, realizar las reconstrucciones de los eventos usando las herramientas de aerie destinadas a dicho fin. Esto generará, como elemento de salida, el insumo principal de sedeb, el cual es un archivo formato Root con eventos reales reconstruidos detectados por HAWC

sedeb facilita al usuario la manipulación de los arhivos que contienen eventos de HAWC. Permite sintetizar en una GUI, tareas que se realizan de manera "manual" bajo la linea de comandos, sin llegar a ser un susbstituto a la herramienta original, por ser un *front-end* a dichas aplicaciones. Al recurrir a las capacidades de las bibliotecas de Qt, se incorporan utilidades que facilitan al usuario la manipulación de información y concretan las tareas repetitivas, un ejemplo de ello es la interface de generación de cortes y su guardado para uso futuro, sin esta utilidad el usuario que diseña algún corte por muy sencillo o complicado que este sea, tendría que ingresarlo nuevamente en cada sesión en la que lo requiera; al tener la funcionalidad de almacenar esta clase de información el usuario solamente tiene que seleccionar el corte almacenado y este será desplegado y estará listo para ser usado nuevamente.

sedeb está desarrollado (cada vez mas estrechamente) con el *framework* de aerie, esto significa que no es una aplicación *stand-alone* (solitaria, independiente), con ello se asegura que el desarrollo y cambios que se van realizando al interior de la colaboracion de HAWC, se vean reflejados directamente en la funcionalidad del programa, sin que necesariamente estos tengan que ser realizados por el programador de la aplicación que usa el framework, en este caso sedeb. La integración de la geometría del detector, las posiciones de los tanques y de los pmt. se ha realizado mediante las clases respectivas definidas en aerie.

sedeb resuelve la necesidad de obtener imágenes, *plots* ó histogramas de manera rápida y sencilla (unos cuantos *clicks* con el ratón) [8] son almacenados en archivos que pueden ser usados a discreción.

Al poder acceder a los datos en los archivos se puede generar un panorama más claro y completo sobre qué buscar y en qué lugar se puede encontrar.

Es una herramienta que ayuda al usuario en el primer acercamiento para la realización de los análisis de datos.

En el diseño y desarrollo de sedeb se ha procurado mantener una sintaxis de

codificación clara que permita al programador un fácil mantenimiento del desarrollo.

Se tiene documentado, en la medida de lo posible, el código fuente, para que facilite el proceso de constante desarrollo, con esto se pretende que la colaboración y personas interesadas en dar seguimiento, tengan los elementos documentales suficientes para entender el funcionamiento de la herramienta. Para ello se ha optado por utilizar el esquema de documentación propuesto por Doxigen [9]. El documento que ésta herramienta genera y mantiene actualizada sobre la información de las clases e implementaciones de sedeb se encuentra en el apéndice D.

sedeb permite cargar archivos en formato Root generados por dos herramientas que provee aerie:

- offline-reconstructor
- xcdf-root

Los archivos generados por la primera herramienta permiten una visualización detallada usando la geometría del detector, mientras que la segunda herramienta no. En ambos casos es posible generar histogramas 1D y 2D mediante los *widgets* que las bibliotecas de Qt proveen.

Es posible generar gráficos X-Y mediante las clases qcustomplot [10] implementadas en sedeb.

Se pueden cargar uno ó varios archivos del mismo tipo, al mismo tiempo; o pueden cargarse bajo la demanda del usuario.

Posee una interfaz de configuración que almacena las variables que requiere el programa para un correcto funcionamiento. Las variables, inicialmente, son tomadas de las variables de entorno, si estas han sido previamente inicializadas con el framework; pueden ser modificadas a voluntad bajo la responsabilidad del usuario.

Posee un editor de cortes de selección y una lista que se llena con el resultado de la aplicación de los cortes. Mediante un cuadro de diálogo es posible seleccionar cortes previamente guardados ó si se prefiere guardar nuevos cortes.

Es posible almacenar los gráficos, ya sea de manera individual ó toda una colección de los mismos. Actualmente se tiene: geometría del detector, tanques y pmt; histograma 1D y histograma 2D. La geometría del detector es guardada en formato de archivo png mientras que los histogramas son almacenados en formato eps.

3.1.1. Event Display en otros proyectos

Un event display debe permitir al usuario tener una imagen panorámica clara de una colección de datos, es por esto que los experimentos, en general, poseen herramientas computacionales de visualización de datos, ya sean propietarias o de propósito general; un *Event Display* no necesariamente es un producto sofisticado. Se pueden construir visualizadores de datos a partir de hojas de cálculo simples con alguna implementación que les permiten generar la interface con la adquisición de datos, programas realizados en diversos lenguajes de programación que despliegan pantallas con información y elementos auxiliares que permitan al usuario interactuar con ella. Conforme la complejidad del experimento y las necesidades de visualización son más exigentes, es entonces cuando una herramienta propietaria de visualización y manipulación de datos es necesaria. Un ejemplo de lo anterior, es el ED (EventDisplay) que muestra los datos tomados por los detectores de superficie y de fluorescencia del observatorio Pierre Auger.

En la figura 3.1 se muestra una captura de pantalla del **ED** del proyecto Auger.

3.2. Diseño

Las etapas realizadas en el proceso de diseño de sedeb, fueron las siguientes:

- Propuesta de solución y discusión al interior de la colaboración, para la visualización de datos mediante una GUI;
- Generación de un mapa mental con la lluvia de ideas;
- Generación de bloques de pseudocódigo;
- Selección de herramientas de desarrollo;
- Selección de ejemplos existentes y conocidos para reproducir y validar resutados

3.2.1. Elementos basicos de sedeb

sedeb requiere acceder a los valores contenidos en archivos en formato Root, (actualmente se está trabajando en la implementación con los archivos XCDF), no solamente para la apertura del archivo en sí, también requiere acceder y recorrer los

valores contenidos el los mismos, para ello cada sistema de archivos tiene su propia clase y estructura, sus métodos de acceso, lectura y escritura, navegación y manipulación de información; para el caso concreto de la manipulación de la información contenida en los archivos Root, con los que trabaja sedeb, se ha diseñado una clase llamada `RootFileFormat`, esta clase permite acceder a los diferentes *Branches* de un archivo Root reconstruido mediante las herramientas de HAWC contenidas en aerie.

`RootFileFormat` está diseñado para no ser dependiente de la plataforma de desarrollo; está desarrollado utilizando única y exclusivamente C++ y las bibliotecas de Root con su implementación para C++. Se ha evitado la llamada a bibliotecas ajenas a este conjunto, con el fin de mantener lo más simple posible el desarrollo. Esto se ve reflejado en funciones miembro que reciben parámetros y arrojan valores con tipos estándar ó tipos y clases definidos en la clase misma. (El diagrama 5.1 muestra la relación de clases con sedeb.)

El diagrama 3.2 muestra, de manera sintética el flujo de trabajo con los datos contenidos en un archivo formato Root; se ve como todo parte desde el punto en que el usuario provee el archivo, pasando por las validaciones y terminando en las representaciones tanto textuales como gráficas que cada uno de los componentes de sedeb realiza sobre ellos.

3.2.2. Organización de las componentes de sedeb

sedeb está organizado de la siguiente manera. Consta de una ventana principal que actúa como contenedora de todos los objetos que constituyen la aplicación completa (widgets). Los widgets son objetos abstractos que representan graficamente distintos elementos, permiten al usuario interactuar con las funcionalidades de la aplicación, en este caso con sedeb. Ejemplo de estos widgets son las "barras deslizables", botones, cajas de texto, display tipo led, selectores de número de precisión simple o de punto flotante, diales, brújulas, selectores de fechas, etiquetas, botones tipo radio, cajas de selección múltiple ó exclusiva, entre otros más, una amplia referencia sobre los widgets de Qt se encuentra en [11].

Al ser representaciones gráficas de funcionalidades particulares en sedeb, los widgets se han distribuido en diferentas partes de la ventana principal; la distribución gráfica de la aplicación permite colocar los widgets dependiendo de la funcionalidad

que se esté accediendo en sedeb.

La ventana principal contiene una barra de menú ó menú principal, un panel de navegación de evento y un paginador por pestañas, este último tiene definidas tantas pestañas como subdivisiones en la organización de la información que se muestra al usuario; los detalles sobre el paginador se encuentran en 3.2.2.2. Estos tres elementos permanecen en una misma ventana y son siempre visibles al usuario, sus caracterísitas son accesibles en cualquier momento.

El menú principal está diseñado siguiendo la estructura que emula la distribución de funcionalidades, nombres de acciones y comportamientos similares a los encontrados habitualmente en una GUI, esto solamente es sustentado por el hecho de que el usuario promedio de computadoras está habituado a encontrar los elementos de la barra de menú en posiciones y nombres bien definidos[1]

La barra de menu principal se encarga de contener a las acciones que realizan la operacionas báse de sedeb, como la carga de archivos, la limpieza de memoria, operaciones que afectan el comportamiento de la aplicación (como la configuración de la misma), así como la de contener todas las herramientas secundarioas ó auxiliares (como el convertidor de formatos).

En la barra de menú se encontrará la siguiente distribución (los nombres de las acciones de los menus aparecen en inglés).

Acción principal	**Acciones asociadas**
File	Open Files, Clear Data, Save, Quit.
Edit	Cuts, Preferences
Tools	Converting tools
Help	About

Cuadro 3.1: menú principal de sedeb

Una descripción detallada de lo que realiza cada una de las acciones asociadas en el menú está documentada en el apéndice C.

[1]Un ejemplo de esto es que la barra de menú, en una aplicación, se encuentra en la parte superior de la ventan principal y la abarca completamente a lo ancho, el primer elemento del menu está relacionado con las operaciones de archivo y al final del menú aparece la opción de Ayuda.

3.2.2.1. Panel de navegación

Este panel está constituido por una etiqueta que lo identifica y una caja de selección. Lo que el panel permite es la realización de la selección y esta se produce introduciento enteros en un rango que está definido entre 0 y el total de eventos cargados al momento.

3.2.2.2. Paginador por pestañas

Este widget es la parte fundamental de sedeb, en el se distribuyen subconjuntos de widgets y funcionalidades en cada pestaña, dependiendo de la categoría a la que la información que se desea desplegar pertenezca, tiene las siguientes categorías ó pestañas [2]:

- General Event Info
- Detailed Event Info
- Simple Histograms
- Simple Graph

Cada una de las pestañas anteriores puede ser accedida en cualquier momento durante la ejecución de la aplicación.

La utilidad del paginador por pestañas es la de contener información y funcionalidades específicas que pueden ser aplicadas para facilitar al usuario la comprensión de los datos que se están analizando. El usuario puede interactuar con la información desplegada en el paginador por pestañas (copiar, pegar, guardar). La elección del paginador por pestañas para organizar la información se realizó en base a los siguientes criterios:

- Flexibilidad: permite agregar, quitar ó modificar pestañas del widget es sencillo y rápida
- Atractivo para el usuario: permite reorganizar las pesañas a voluntad, arrastrando y soltando o navegar entre pestañas mediante combinaciones de teclas

[2] se preservan los nombres en inglés para guardar consistencia con la aplicación

- Claridad: Organiza y despliega información de manera clara, amigable
- Compacidad: no ocupa espacio adicional sin importar el número de pestañas que se tengan definidas, además de mantenerlas agrupadas

En el diseño se ha tomado en cuenta el esquema de *SIGNALS-SLOTS* [11] (Señales y Ranuras) de Qt, que permite interconectar los widgets entre sí para lograr que se comuniquen, ello facilita el trabajo de programación e implementación.

3.3. Obtención de datos

En el sitio del observatorio se aquieren los datos de los eventos que son detectados, estos son almacenados y poseriormente son transferidos a los servidores que permiten un fácil acceso a los miembros de la colaboración. De esta manera el usuario puede conectarse a alguno de estos servidores y transferir el ó los archivos deseados; usualmente están almacenados en una estructura por directorios nombrados por fecha (añomesdía).

3.4. Representación y visualización de datos

La estructura de datos, en la cual se almacenan los eventos generados por la colaboración HAWC, ha estado evolucionando y optimizándose para ser cada vez más pertiente con el experimento.

La visualización de eventos permite tener una imagen clara, con la capacidad de acceder de manera compacta a su información, lo que se traduce en acceder a la mayor cantidad de información relacionada a un objeto, sin tener que mostrarla de una sola vez en un mismo lugar, pero que esta pueda ser alcanzada con funcionalidades simples y transparentes por el usuario: menus secundarios, globos de texto emergentes, *tool tips*.

Los eventos reconstruidos y almacenados en archivos formato Root, poseen información detallada de un evento detectado por HAWC. Las variables físicas que dicho evento contiene, tales como la cantidad de tanques activados, cuantas veces fueron activados los pmt, tiempos de activación, fecha, posiciones del núcleo y core

del chubasco (en diferentes ajustes), ajustes a los planos del chubasco, entre otras mas.

Para lograr cargar los archivos Root dentro de seded, se ha escrito una clase en C++, `RootFileFormat` la cual permite acceder a la estructura de un archivo en formato Root y cargar su contenido en memoria y hacerlo accesible a las diferentes partes de sedeb. Esta clase permite interactuar con las capacidades de Root y tomar en cuenta los requisitos particulares del usuario.

Para realizar lo anterior es necesario especificar la dirección de la memoria mediante el nombre de la variable usada en la estructura de archivos de Root y luego enlazarla con el nombre de la variable que se usará en sedeb. Esta es una de las partes fundamentales de sedeb ya que en el diseño se ha tomado la precaución de mantener lo más accesible posible la manera en que se hace esta asignación y relación. los detalles sobre la manera de realizar este enlace se presenta en el apéndice **??**

3.5. Carga archivos

Root utiliza un esquema en el cual separa en categorías los conjuntos de variables que el usuario requiere almacenar, con una jerarquización que parte de lo general a lo particular en un sistema de archivos al estilo de una base de datos. La jerarquía superior es llamada Árbol (Tree), este a su ves contiene Ramas (Branches) y este contiene Hojas (Leafs); puede haber tantas de cada una como se requiera.

Los detalles de la estructura de los archivos Root que sedeb carga, se presentan en Apéndice A. En este proceso RoofFileFormat provee a sedeb de las direcciones de memoria necesarias para almacenar las referencias a los valores al cargar cada evento. Cada petición por parte del usuario para actualizar la información de un evento en el archivo, es reflejada tanto en el panel de navegación como en el paginador por pestañas, llenando con los valores ó representaciones de los mismos los campos asignados en cada pestaña del paginador; en el caso de los historgramas estos corren sobre todos los valores de la rama seleccionada en su respectiva pestaña.

sedeb permite, con los datos cargados, que el usuario tenga una manera de ver y manipular la información contenida en los archivos que está utilizando, todo ello mediante la interfaz que constituye sedeb.

3.6. Representación gráfica de eventos

La representación gráfica de un evento de HAWC en sedeb utiliza el recurso del dibujo de geometrías primitivas en la perstaña *General Rvent Info*, plasmando una instantánea del evento reconstruido en el momento que HAWC lo registró, (la fecha exacta del evento queda registrada en variables como mjd, gpsSecond, gpsNanoSecond, descritas en el Apéndice A).

El poseer las coordenadas y valores reportados por los dispositivos que conforman el detector de HAWC proporciona el insumo principal para la generación de esta instantánea gráfica que se muestra. El área de dibujo de *General Event Info* llamada Canvas Detector representa la geometría del detector de HAWC con el arreglos de tanques y pmt. Las opciones que aparecen en esta ventana y que no pertenecen al canvas permiten al usuario la manipulación de algunos elementos gráficos que ayudan a una visualización conveniente para ciertos aspectos (ocultar los tanques ó pmt ó ambos, solo mostrar, ajuste de la representación de valores mediante el cambio de radios y colores).

3.7. Canvas Detector

Este objeto es diseñado tomando en cuenta que representará gráficamente al evento seleccionado, de tal manera que debe de plasmar en la instantánea que genera de manera sencilla y directa la mayor cantidad de información posible. Para lograr esto se utilizan diversas funcionalidades que Qt provee, como las *tool tip* asociado a cada elemento del Canvas Detector y que se muestran solo cuando se posiciona el puntero del ratón sobre el objeto en cuestión.

Los elementos que constituyen al detector se obtienen directamente desde la clase detector de aerie, esta contiene la información necesaria para poder reproducir la geometría del detector así como también la información relacionada a cada elemento (por ejemplo, su nombre ó número de asignación).

Al integrar al desarrollo el framework de aerie, cualquier modificación al detector y sus componentes que sea realizado de manera oficial, se verá reflejada en esta clase y evitará la tarea de mantenimiento, duplicidad de trabajo y asegurará consistencia con el desarrollo que la colaboracion de HAWC va realizando.

Los *tool tip* que se han diseñado para Canvas Detector por el momento incluyen el nombre de cada objeto y se tiene contemplada la incorporación de otros datos conforme se discutan y aprueben las peticiones recibidas por grupo de software de la colaboración, esto es tema de discución para el grupo de Software de la colaboración.

La representación gráfica de los objetos del detector de HAWC es sencilla. El detector está representado exclusivamente por los tanques y sus pmt asociados, estos están definidos con circunferencias (de mayor diámetro para los tanques). Cuando un evento registra actividad en los pmt estos últimos son dibujados con un color distinto semitransparente, de tal manera que cuando existe una recurrencia a cierta posición el color de este objeto se va obscureciendo, lo que permite un contraste visual con el resto de los elementos que lo constituye.

3.7.1. Desarrollo de Canvas Detector

Para su desarrollo se usan las clases QGraphicsScene y QGraphicsView, que son parte del QGraphics View Framework de Qt[11].

La clase QGraphicsScene provee una superficie para manipular un gran número de elementos gráficos en 2D. La clase es contenedor para los QGraphicsItems. Es usada en conjunto con QGraphicsView para la visualización de elementos gráficos, tales como lineas, rectángulos, texto, ó inclusive elementos propietarios, en una superficie 2D.

QGraphicsScene también provee la funcionalidad que permite determinar eficientemente la ubicación de elementos y cuales de ellos serán visibles dentro de un área arbitrariamente designada en la escena[11].

La clase QGraphicsView provee un widget para desplegar el contenido de QGraphicsScene. QGraphicsView visualiza el contenido de una QGraphicsScene en un región de visualización desplazable[11].

La implementación de la construcción tanto del detector como de los eventos dentro del Canvas Detector supone una transformación de coordenadas, dado que el sistema de referencia de Qt es absuluto, el origen es localizado en la esquina superior izquierda de la escena.

El detector posee un sistema de coordenadas que situa el origen en la esquina inferior izquierda, la transformación viene dada como:

$$P(x, y) = ((x - r)k1, (yMax - y - r)k2) \tag{3.1}$$

donde (x, y) son las coordenadas de punto que se desea representar, $yMax$ es el máximo valor de y localizado en el detector, $k1$ y $k2$ son las constantes de proporcionalidad, las cuales no necesariamente son iguales debido a que la geometría del Canvas Detector es rectangular y los lados cambian sus dimensiones bajo demanda, r es el radio del objeto en cuestion. La inclusión de r es una corrección necesaria para mantener consistencia con el origen de cada elemento geométrico, situándolo en el centro geométrico del elemento. Qt maneja el origen de todos sus elementos geométricos en la esquina superior izquierda de una caja rectangular que lo contenga, este caso particular un solo parámetro es suficiente, ya que nuestros elementos son circunferencias de radio r.

El Canvas Detector recibe una colección de puntos que es transformada y luego representada graficamente en el espacio destinado al dibujo de las primitivas dentro de sedeb; a cada objeto se le asigna una propiedad específica (color, radio, posición) dependiendo de su naturaleza.

La única manera de interactuar con los elementos dentro del Canvas Detector es mediante el posiciónamiento del cursor del mouse sobre el objeto, eso despliega un *tool tip* con la información relacionada al mismo.

Los controles adicionales en General Event Info sirven para selecciónar el comportamiento visual del Canvas Detector, estos son:

- Mostrar, Ocultar: pmt;
- Mostrar, Ocultar: Tanques;
- El radio representa;
- El color representa.

Cada elemento que es desplegado dentro del Canvas Detector es independiente y es dibujado con cada modificación del Canvas Detector, esto se ve reflejado en el tiempo de procesamiento al generar la imágen de cada evento. La cantidad de objetos gráficos contenidos en el Canvas Detector es proporcional al tiempo que se tarda en generar la imágen final correspondiente.

Las coordenadas de los tanques y pmt, así como las dimensiones y los nombres con los que se identifican son obtenidos mediante la clase `detector` de aerie.

3.8. Detailed Event Info

Este elemento del paginador por pestañas es un cuadro de texto en el cual es vaciada la información correspondiente, única y exclusivamente, al evento actualmente seleccionado. Despliega en un formato de texto simple las variables y los valores relacionados al evento actual. No permite edición directa en pantalla, lo cual evita modificar accidentalmente algún valor ó información. Permite la copia, mediante el proceso de selección de bloques de texto, (ya sea mediante el teclado ó con el ratón) para poder ser utilizados en alguna otra aplicación fuera de sedeb. A cada nueva selección de evento la información es actualizada en esta ventana.

El diseño de esta parte este elemento está basado en la construcción de una pestaña que contiene un editor de texto de Qt, construido con la clase `QtTextEdit Widget`, esta clase, una vez implementada permite la manipulación de flujos de texto en diferentes formas (`string stdout`, Qt string, caracteres). El llenado del elemento se hace llamando a la funciones miembro de `RootFileFormat`:

- `GetEventInfo(Event *);`
- `GetcoreGuessInfo(coreGuess *);`
- `GetcoreGaussInfo(coreGauss *);`
- `GetplaneFitInfo(planeFit *,int);`
- `GetplaneFitOddInfo(planeFitOdd *,int);`
- `GetplaneFitEvenInfo(planeFitEven *,int);`
- `GetlhFitInfo(lhFit *);`
- `Getcompactness40Info(compactness40 *);`
- `GetstdCutsInfo(stdCuts *);`

- `GetXCDFTrigInfo(RootTrig *);`

Estas funciones miembro, públicas de `RootFileFormat`, son independientes de Qt y pueden ser implementadas en programas ajenos a sedeb, su valor de retorno es una cadena de texto de la salida estándar, permiten una rápida modificación en el formato de salida.

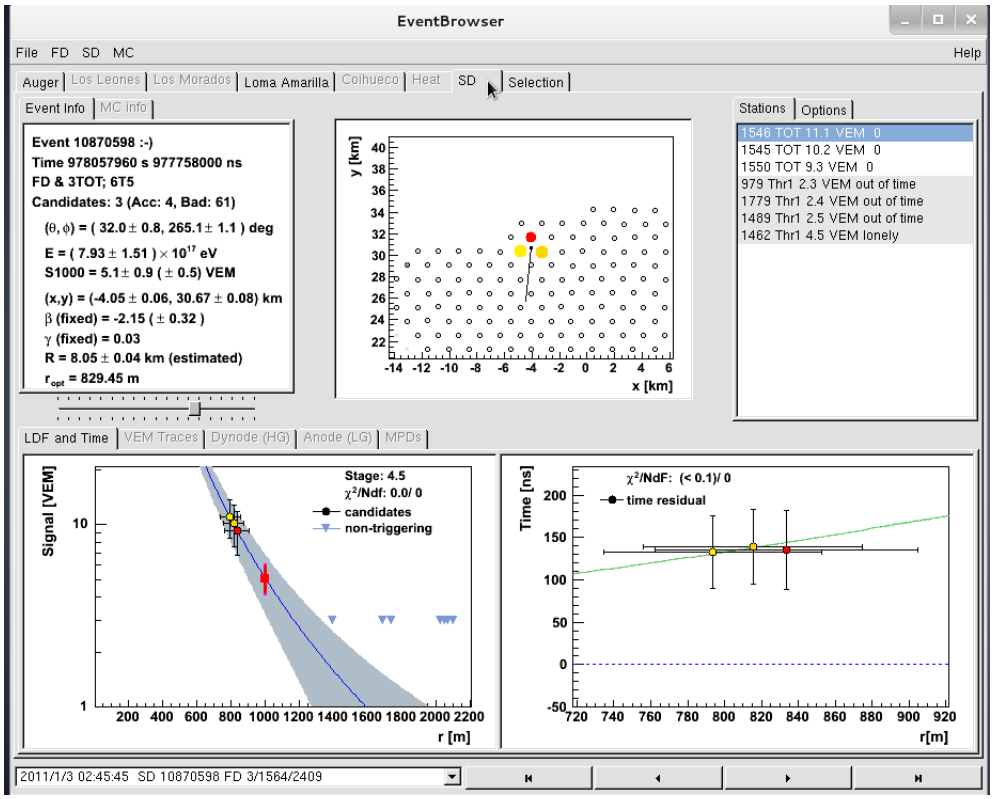


Figura 3.1: Captura de pantalla del ED del Proyecto Auger

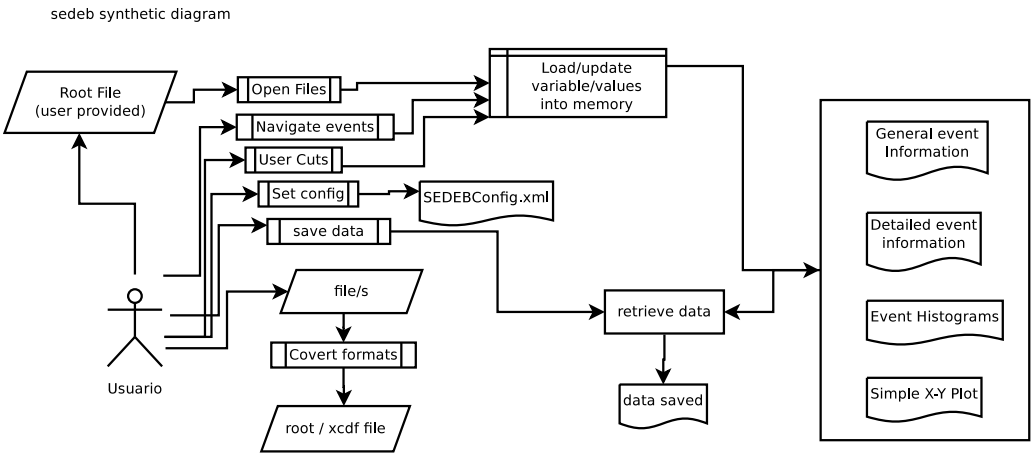


Figura 3.2: Diagrama sintético de sedeb

CAPÍTULO 4

ASPECTOS TÉCNICOS DEL DESARROLLO

4.1. Datos técnicos de sedeb

Nombre de la aplicacion	sedeb
Nombre del desarrollador	Cederik León De León Acuña
Lenguaje empleado	C++
Bibliotecas empleadas	Qt 4.8 y ROOT 5.34
Total de archivos .cpp	15
Total de archivos .h	21
Total de archivos .ui	6
Total de archivos auxiliares	7
Número total de lineas de código	24858

Cuadro 4.1: Datos técnicos de sedeb

4.2. Sistema Operativo

En este trabajo se empleó el sistema operativo basado en Unix, Linux, en su Distribución GNU Linux/DEBIAN version 7.0.0 [12]. Linux es un sistema operativo para

computadoras, de alto rendimiento y confiblilidad, basado en Unix. Fue programado, inicialmente por el computologo finlandés Linus Benedict Torvalds y su primera versión fue publicada en internet, de manera gratuita a finales de 1991. Tiene sus orígenes en el sistema operativo Minix del Dr. Andrew S. Tanembaum (minix es un Mini Unix que programó el Dr. Tanenbaum desde primeros principios, como una herramienta de apoyo, didáctica, útil en su curso de sistemas operativos en la Universidad de Amsterdam en Holanda).

Linux provee al usuario de capacidades de cómputo multitarea y multiusuario, además de que en su desarrollo han intervenido cientos de desarrolladores de todo el mundo. Por añadidura se ha convertido, casi de manera natural, en un sistema multiplataforma, lo que permite tener disponible todas las caracteristicas de Linux en diferentes arquitecturas de computo.

Dada la filosofia inicial del sistema operatvo en cuestion, facilita al usuario de cómputo doméstico un sistema operativo, robusto y confiable pero al mismo tiempo accesible; es posible instalarlo en una gran variedad de equipos de cómputo y es posible compartir aplicaciones y/o distribuirlas.

Ya que el sistema operativo Linux es de libre acceso y distribución, su costo es mínimo, casi limitandose al costo de los equipos de cómputo relacionados enteramente al hardware.

Linux es un sistema operativo que permite la interconexión de diversos equipos entre si de manera transparente, esto se ve reflejado en la posible configuración de equipos que trabajen de manera distribuida, permitiendo la obtención de resultados en una menor cantidad de tiempo. Linux es un sistema que es practicamente invulnerable a los ataques de los virus informáticos, lo que da una mayor estabilidad y seguridad a los datos de los usuarios.

El proceso de desarrollo de sedeb presentado en este trabajo se realizó en plataformas de cómputo con GNU/Linux Debian.

4.3. Lenguajes de programación

Muchas de las herramientas computacionales empleadas en el cálculo y/o análisis de datos en la física están desarrolladas en lenguajes de programación que explotan

al máximo las capacidades tecnológicas existentes al momento de su producción.

Diversos lenguajes de programación son empleados, dependiendo de las necesidades y naturaleza de los problemas a resolver. Un factor importante que es considerado para la elección de los lenguajes de programación en la producción de programas, son las habilidades personales del programador y el gusto ó afinidad (sujetiva) que profesa por uno u otro lenguaje en particular.

En casos donde las necesidades de cálculo son excepcionalmente demandantes, muchos programadores prefieren lenguajes de programación orientados al computo numérico, como ejemplo podemos referirnos a FORTRAN [13].

En el desarrollo de sedeb se opta por el lenguajes de programación C++.

4.3.1. C/C++

El lenguaje C es una herramienta de programacion necesaria e indispensable para todo aquel que desea realizar programación estructurada, de gran portabilidad y fácil acceso. El lenguaje C tiene sus orígenes en los lenguages BCLP y B hacia finales de 1960, la derivación del lenguaje B hacia C fue desarrollada por Dennis Ritchie en los Laboratorios Bell. En sus inicios, C fue popular dentro del circulo de diseñadores y programadores de sistemas operativos, quienes lo utilizaron para sus productos, siendo un ejemplo el sistema operativo UNIX, el cual fue programado en el lenguaje C. Dado que C es un lenguaje independiente del hardware y ampliamente disponible, las aplicaciones que estan escitras en C pueden ejecutarse con poca o ninguna modificación en una amplia gama de sistemas distintos de cómputo [14].

Conforme se elabora una herramienta computacional ó entorno de desarrollo, los lenguajes de programación pueden verse rebasados, en cuanto a la flexibilidad y/o la adecuación a un problema dado. En el caso de la física de altas energías y en especial de los programas y paquetes de cómputo, el lenguaje C (refiriendonos al "ANSI C") pudiera quedarse rezagado si no fuera por el "super-conjunto" de C, llamado C++ [15] desarrollado por Stroustrup. Este lenguaje permite a los programadores desarrollar programas de cómputo bajo el paradigma de la "Programacion Orientada a Objetos", generando con ello piezas de software que pueden ser reutilizadas y aplicadas de diversas maneras en un proyecto.

4.4. Bibliotecas empleadas en el desarrollo

4.4.1. Bibliotecas Qt

Qt es un entorno de desarrollo basado en C++, permite la programación de aplicaciones para computadora, está fuertemente orientado a la construcción de GUI, para ello proporciona un conjunto de herramientas completo, entre ellas Qt Creator [16].

Qt es usado en los siguientes paquetes Autodesk Maya, Adobe Photoshop Elements, OPIE, Skype, VLC media player, VirtualBox y Mathematica y además por las siguientes empresas e instituciones, la Agencia Espacial Europea, DreamWorks, Google, HP, Lucasfilm, Panasonic, Philips, Samsung, Siemens, Volvo y por Walt Disney Animation Studios[17].

Qt Creator es una herramienta que tiene como propósito la administración del desarrollo y la codificación de las aplicaciones en las que se esté trabajando, también es usado como un constructor de GUI muy poderoso.

Una de las cualidades principales de Qt, es la de ser *cross-platform*, esto significa que la portabilidad del desarrollo entre sistemas operativos y plataformas diversas se realiza de forma transparente.

Qt está liberado bajo la licencia GPL y puede ser usado de manera gratuita, en caso de que el desarrollo sea no comercial[18].

4.4.1.1. Uso de Qt en sedeb

El entonrno de desarrollo proporcionado por las herramientas de Qt fue elegido debido a lo siguiente:

- el autor del presente trabajo está familiarizado con la herramienta
- es gratuita
- accesible (puede ser descargada desde el sitio oficial `http://qt.digia.com`).
- posee una documentación exhaustiva
- es actualizada permanentemente
- es portable entre plataformas

- Es adaptable a cualquier gusto de usuario, sus elementos gráficos o widgets tendran el aspecto que mas le agrade al usuario, esto permite una mejor aceptación del programa
- posee una extensa colección de funciones y métodos, lo cual facilita y reduce la codificación al programador
- permite la reimplementación de sus métodos y funciones
- el diseñador de interfaces gráficas (qdesigner) reduce el tiempo de programación;
- posee un amplia colección de widgets (elementos gráficos) y su esquema de trabajo es *drag and drop* (arrastrar y soltar)

4.4.2. Bibliotecas ROOT

ROOT [7] es un entorno de trabajo para el procesamiento de datos, nacido en el CERN, en el corazón de los grupos de investigación de física de altas energías. Diariamente, miles de físicos utilizan aplicaciones ROOT para el análisis de sus datos ó para realizar simulaciones.

La estructura de ROOT permite guardar los datos en un formato binario comprimido llamado Archivo ROOT. La estructura de este formato de archivo permite que los datos almacenados en él sean fácil y rapidamente accesdidos, casi sin importar la cantidad de datos almacenados.

Este formato permite que los datos guardados se puedan acceder, ya sea en una computadora de escritorio, desde la web y mediante un sistema de distribución de archivos masivo como por ejemplo GRID. Esto se logra mediante la distrubución en forma de árbol que hace ROOT en sus archivos, permitiendo encadenarlos y acceder a ellos como si fueran un solo objeto.

En lo relacionado al proceso de los datos, ROOT cuenta con herramientas matemáticas y de estadística poderosas que pueden ser usadas para operar en los datos del usuario. ROOT dispone de toda la capacidad de una aplicación tipo C++ y de proceso en paralelo, lo que permite simular sistemas muy complicados.

Entre los atractivos y características de ROOT es la generación de histogramas, gráficos de dispersión, funciones de ajuste. Los gráficos generados por ROOT pueden

ser ajustado en tiempo real mediante el uso del puntero del "ratón". Además de que pueden ser guardados en formato PDF ó cualquier otro formato gráfico.

Se puede hacer uso de la CINT C++ (intérprete de C++) ó de Python para crear sesiones interactivas y para escribir macros, también para compilar un programa para que pueda hacer uso de todas las capacidades de cómputo, sin interpretación previa, lo que se refleja en que dicha aplicación se ejecute a mayor velocidad. En ambos casos, ROOT permite construir interfaces gráficas.

CAPÍTULO 5

RESULTADOS Y VALIDACIÓN

Para validar los resultados que se obtienen con el uso de sedeb, se han realizado pruebas y comparaciones entre resultados, obtenidos previamente con herramientas diversas, programas propietarios ó implementaciones para generar gráficos, histogramas y despliegue de información que facilitan el análisis a partir de los datos generados por HAWC.

Los eventos reconstruidos de HAWC pueden ser almacenados en un archivo en formato de Root con una estructura definida, el proceso y herramientas de generación de estos archivos son procesos implementados y distribuidos dentro del conjunto de herramientas o *framework* de aerie. El proceso de lectura y despliegue de la información de los archivos Root de HAWC dentro de sedeb, se realiza utilizando la clase y funciones contenidas en `RootFileFormat`, estas están vinculadas con las propias clases de Root, el diagrama 5.1 muestra la relación de clases con sedeb.

Esto garantiza un correcto acceso, lectura y almacenamiento de los datos.

Se realizaron pruebas para validar que los datos contenidos en el archivo Root fueran consistentes con los que se muestran al usuario en sedeb, abriendo un mismo archivo en formato Root, selecciona el mismo evento y comparando los resultados arrojados por ambos programas, la validación fue exitosa.

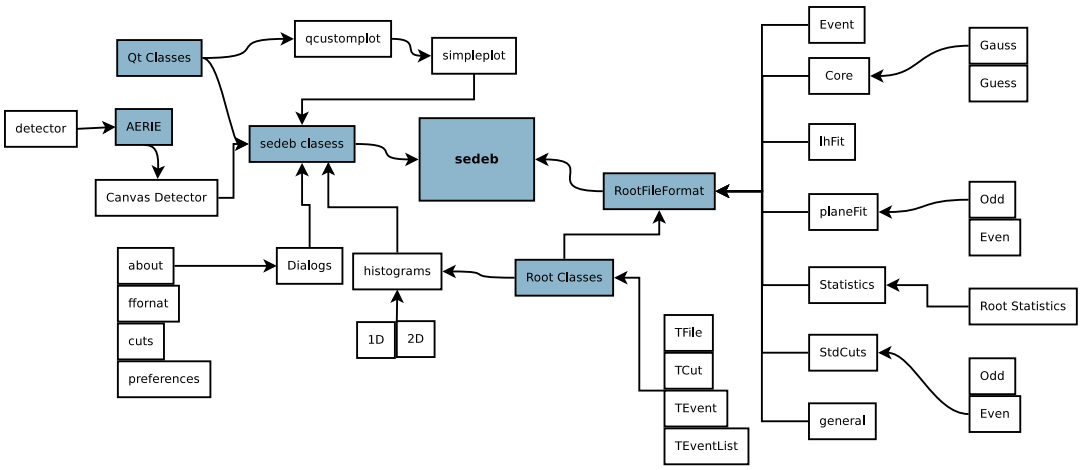


Figura 5.1: Relacion entre clases usadas en sedeb

CAPÍTULO 6

CONCLUSIONES

Se ha diseñado, desarrollado e implementado una herramienta computacional que permite la visualización de archivos que contienen eventos reconstruidos para el proyecto HAWC.

Se ha publicado una nota técnica al interior de la colaboración internacional HAWC como producto del presente trabajo.

El diseño actual de sedeb permite que se siga desarrollando, mejorando y manteniendo para que se adapte de manera eficiente y pertinente a las necesidades de visualización y manipulación de eventos de la colaboración internacional de HAWC.

Se ha mostrado que las herramientas computacionales científicas pueden ser construidas utilizando recursos que pueden ser adquiridos libremente y que además prueban ser estables y confiables.

Lo anterior es un ejemplo de diseño y desarrollo, no solo para actividades relacionadas con aspectos científicos de adquisición y visualización de datos en experimentos ó simulaciones, sino también en ambientes de producción ó maquila pó cualquier actividad industrial ó comercial que involucre monitorear o visualizar colecciones de datos.

El seguimiento del desarrollo se realiza mediante las reuniones semanales del grupo de software de la colaboración HAWC, donde se reciben y discuten peticiones y recomendaciones de los participantes para nuevas implementaciones ó mejoras que posteriormente se incorporan.

6.1. Trabajo futuro

Se continuará manteniendo y desarrollando sedeb en base a las espectativas y necesidades de la colaboración. Lo anterior se logrará dandole continuidad a la asistencia y participación en las reuninoes de la colaboración HAWC, tambien promoviendo y difundiendo las capacidades del software, esto involucra el establecimiento de cursos y talleres que expliquen la utilización y funcionamiento de sedeb. De esta manera, además de promover el desarrollo realizado, se busca la retroalimentación por parte del usuario final e inclusive la incorporación de desarrolladores que puedan aportar mejoras al mismo.

BIBLIOGRAFÍA

[1] Colaboración HAWC. Sitio oficial de hawc. `http://hawc.umd.edu/`.

[2] A.U. Abeysekara, R. Alfaro, C. Alvarez, J.D. Álvarez, R. Arceo, et al. Sensitivity of the High Altitude Water Cherenkov Detector to Sources of Multi-TeV Gamma Rays. 2013.

[3] Colaboración HAWC. Colaboración hawc. `http://hawc.umd.edu/collaboration`.

[4] CERN, S. Southworth, and Gordon. Fraser. Achievements with antimatter. `http://cern-discoveries.web.cern.ch/CERN-Discoveries/Courier/Intro.htm`.

[5] W.-M. Yao, C. Amsler, D. Asner, R.M. Barnett, J. Beringer, P.R. Burchat, C.D. Carone, C. Caso, O. Dahl, G. D'Ambrosio, A. DeGouvea, M. Doser, S. Eidelman, J.L. Feng, T. Gherghetta, M. Goodman, C. Grab, D.E. Groom, A. Gurtu, K. Hagiwara, K.G. Hayes, J.J. Hernández-Rey, K. Hikasa, H. Jawahery, C. Kolda, Kwon Y., M.L. Mangano, A.V. Manohar, A. Masoni, R. Miquel, K. Mönig, H. Murayama, K. Nakamura, S. Navas, K.A. Olive, L. Pape, C. Patrignani, A. Piepke, G. Punzi, G. Raffelt, J.G. Smith, M. Tanabashi, J. Terning, N.A. Törnqvist, T.G. Trippe, P. Vogel, T. Watari, C.G. Wohl, R.L. Workman, P.A. Zyla, B. Armstrong, G. Harper, V.S. Lugovsky, P. Schaffner, M. Artuso, K.S. Babu, H.R. Band, E. Barberio, M. Battaglia, H. Bichsel, O. Biebel, P. Bloch, E. Blucher,

R.N. Cahn, D. Casper, A. Cattai, A. Ceccucci, D. Chakraborty, R.S. Chivukula, G. Cowan, T. Damour, T. DeGrand, K. Desler, M.A. Dobbs, M. Drees, A. Edwards, D.A. Edwards, V.D. Elvira, J. Erler, V.V. Ezhela, W. Fetscher, B.D. Fields, B. Foster, D. Froidevaux, T.K. Gaisser, L. Garren, H.-J. Gerber, G. Gerbier, L. Gibbons, F.J. Gilman, G.F. Giudice, A.V. Gritsan, M. Grünewald, H.E. Haber, C. Hagmann, I. Hinchliffe, A. Höcker, P. Igo-Kemenes, J.D. Jackson, K.F. Johnson, D. Karlen, B. Kayser, D. Kirkby, S.R. Klein, K. Kleinknecht, I.G. Knowles, R.V. Kowalewski, P. Kreitz, B. Krusche, Yu.V. Kuyanov, O. Lahav, P. Langacker, A. Liddle, Z. Ligeti, T.M. Liss, L. Littenberg, L. Liu, K.S. Lugovsky, S.B. Lugovsky, T. Mannel, D.M. Manley, W.J. Marciano, A.D. Martin, D. Milstead, M. Narain, P. Nason, Y. Nir, J.A. Peacock, S.A. Prell, A. Quadt, S. Raby, B.N. Ratcliff, E.A. Razuvaev, B. Renk, P. Richardson, S. Roesler, G. Rolandi, M.T. Ronan, L.J. Rosenberg, C.T. Sachrajda, S. Sarkar, M. Schmitt, O. Schneider, D. Scott, T. Sjöstrand, G.F. Smoot, P. Sokolsky, S. Spanier, H. Spieler, A. Stahl, T. Stanev, R.E. Streitmatter, T. Sumiyoshi, N.P. Tkachenko, G.H. Trilling, G. Valencia, K. van Bibber, M.G. Vincter, D.R. Ward, B.R. Webber, J.D. Wells, M. Whalley, L. Wolfenstein, J. Womersley, C.L. Woody, A. Yamamoto, O.V. Zenin, J. Zhang, and R.-Y. Zhu. Review of Particle Physics. *Journal of Physics G*, 33:1+, 2006.

[6] Gustavo E. Romero. Introducción a la astronomía y astrofísica de rayos gamma. Presentacion curso impartido en CRyA UNAM, Morelia, 2013.

[7] The ROOT Team. Introducción a ROOT. `http://root.cern.ch/drdupal/content/gentle-introduction`, 1995-2010.

[8] De-León. C, and Cotti, U. and Álvarez. J . sedeb Users guide. Technical note for the HAWC Colaboration, 2013.

[9] Dimitri van Heesch. Doxygen. `http://www.stack.nl/~dimitri/doxygen/index.html`.

[10] Eichhammer. Emanuel . QCustomPlot, a simple to use, modern plotting widget for Qt. Technical document, 2012.

[11] Nokia Corporation. Qt 4.7 : Programming with Qt. Qt documentation, 2008,2010.

[12] Software in the Public Interest, Inc. GNU LINUX DEBIAN. `http://www.debian.org/intro/about`.

[13] Justo R. Pérez Cruz. Notas elementales sobre programación en fotran. *Departamento de Física Fundamental y Experimental Electrónica y Sistemas*, 2008.

[14] H. M. Deitel and P. J. Deitel. *Como programar en C / C++.* Prentice Hall, segunda edition, 1994.

[15] Bjarne Stroustrup. *El lenguaje de programación C++.* Addison Wesley, especial edition, 2001.

[16] Majer, Adam. and Laine, Jeremy. qtcreator - Integrated Development Environment for Qt. GNU Linux Man Page for the Debian project.

[17] KDE Doc Team. *KDE.* KDE, first edition, 2011.

[18] Nokia Corporation. Qt 4.7: Open Source Versions of Qt. Qt documentation, 2008,2010.

APÉNDICE A

ESTRUCTURA DE ARCHIVO ROOT HAWC DE EVENTOS RECONSTRUIDOS

'Tree' principal: Events
'Branch': event

- event.eventID
- event.runID
- event.timeSliceID
- event.gpsSecond
- event.gpsNanosecond
- event.mjd
- event.secondOfDay
- event.nHit
- event.nCh
- event.nTank
- event.eventFlags
- event.triggerFlags
- event.gtcFlags
- event.sumPE
- event.laserTStart

- event.laserTStop
- event.laserLightToTanksStart
- event.laserLightToTanksStop
- event.hit.gridId
- event.hit.tankId
- event.hit.channelIsGood
- event.hit.xPMT
- event.hit.yPMT
- event.hit.zPMT
- event.hit.time
- event.hit.charge
- event.hit.loTOTCharge
- event.hit.hiTOTCharge
- event.hit.rawTime
- event.hit.loTOT
- event.hit.hiTOT
- event.hit.time01
- event.hit.flags
- event.hit.triggerFlags

'Branch': coreGuess

- coreGuess.xCore
- coreGuess.yCore
- coreGuess.zCore
- coreGuess.sigma
- coreGuess.amplitude
- coreGuess.uncertaintiesCalculated
- coreGuess.xCoreUncertainty
- coreGuess.yCoreUncertainty
- coreGuess.zCoreUncertainty
- coreGuess.sigmaUncertainty

- coreGuess.amplitudeUncertainty
- coreGuess.nFit
- coreGuess.status

'Branch': coreGauss

- coreGauss.xCore
- coreGauss.yCore
- coreGauss.zCore
- coreGauss.sigma
- coreGauss.amplitude
- coreGauss.uncertaintiesCalculated
- coreGauss.xCoreUncertainty
- coreGauss.yCoreUncertainty
- coreGauss.zCoreUncertainty
- coreGauss.sigmaUncertainty
- coreGauss.amplitudeUncertainty
- coreGauss.chiSquared
- coreGauss.nFit
- coreGauss.status

'Branch': planeFit

- planeFit.theta
- planeFit.phi
- planeFit.referenceTime
- planeFit.uncertaintiesCalculated
- planeFit.angularUncertainty
- planeFit.timeUncertainty
- planeFit.chiSq
- planeFit.Ndof
- planeFit.tResi
- planeFit.nFit
- planeFit.status

'Branch': lhFit

- lhFit.theta
- lhFit.phi
- lhFit.referenceTime
- lhFit.uncertaintiesCalculated
- lhFit.angularUncertainty
- lhFit.timeUncertainty
- lhFit.chiSq
- lhFit.Ndof
- lhFit.tResi
- lhFit.nFit
- lhFit.status

'Branch': compactness40

- compactness40.radius
- compactness40.compactness
- compactness40.maxPE
- compactness40.nFit
- compactness40.status

'Branch': planeFitEven

- planeFitEven.theta
- planeFitEven.phi
- planeFitEven.referenceTime
- planeFitEven.uncertaintiesCalculated
- planeFitEven.angularUncertainty
- planeFitEven.timeUncertainty
- planeFitEven.chiSq
- planeFitEven.Ndof
- planeFitEven.tResi
- planeFitEven.nFit

- planeFitEven.status

'Branch': planeFitOdd

- planeFitOdd.theta
- planeFitOdd.phi
- planeFitOdd.referenceTime
- planeFitOdd.uncertaintiesCalculated
- planeFitOdd.angularUncertainty
- planeFitOdd.timeUncertainty
- planeFitOdd.chiSq
- planeFitOdd.Ndof
- planeFitOdd.tResi
- planeFitOdd.nFit
- planeFitOdd.status

'Branch': stdCuts

- stdCuts.nHit
- stdCuts.nCh
- stdCuts.nTank
- stdCuts.isSelected

'Branch': stdCutsEven

- stdCutsEven.nHit
- stdCutsEven.nCh
- stdCutsEven.nTank
- stdCutsEven.isSelected

'Branch': stdCutsOdd

- stdCutsOdd.nHit
- stdCutsOdd.nCh
- stdCutsOdd.nTank
- stdCutsOdd.isSelected

APÉNDICE B

SEDEB: CAPTURAS DE PANTALLA

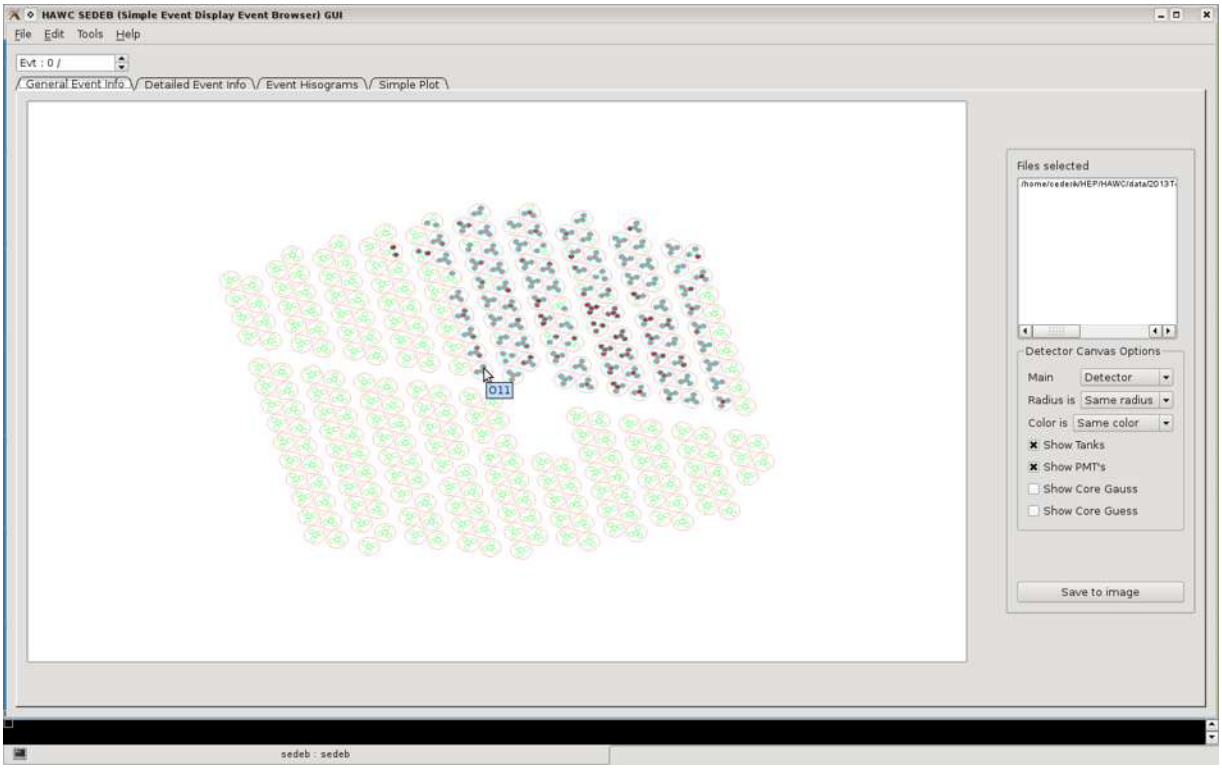


Figura B.1: Pantalla principal de sedeb, donde se muestra la representación del detector y un evento reconstruido

APÉNDICE C

SEDEB: MANUAL DEL USUARIO

sedeb (S imple E vent D isplay E vent B rowser)

for the HAWC Collaboration

Authors:
Juan de Dios Álvarez Romero, Umberto Cotti Gollini, Sir Cederik León De León Acuña.
Universidad Michoacana de San Nicolás de Hidalgo.
Grupo de Física Computacional.
jdalvarez@ifm.umich.mx ucotti@ifm.umich.mx cederik@gmail.com

2013

Table of Contents

Abstract

sedeb is a software tool that allow users to visualize HAWC experiment data files; it's a graphical user interface (GUI), can be used to perform visualization on a collection of events contained in a file or a bunch of files generated by the HAWC experiment. Through it's canvas detector a selected event can be represented graphically with a set of circumferences and colored circles. This tool is capable to show detailed information about an event, make and apply cuts on them, save cuts for further use, make and save simple histograms and plots. Provides a graphical front-end to some converting and reconstructing tools from Aerie framework.

We present in this document the current state of art on the development of the tool, as well a "User guide", with topics covering from getting the software through compilation process to the use of it's components.

keys: visualize, display, browser, event, software

Introduction.

A collection of data can be managed by users in many ways, it implies that the needs to perform a selection varies from every user; sedeb it's mainly designed and programmed as a graphical user interface (GUI) tool, using the Qt4 libraries and the Aerie framework that helps the users to make a first approach to the HAWC data files generated, make a slightly see on data, navigate through millions of events contained in a file or files, search for events that satisfy a desire set of cuts on certain parameters, visualize them graphically, save the results and share them.

The sedeb is intended to helps the users in the analysis process at a very first stage, also helps the user to perform, graphically, some console oriented tasks as file format conversion or reconstruction processes.

The sedeb It's in a continuous development process, it's susceptible to improvements and changes in order to be pertinent to the HAWC collaboration

The sedeb it's not an analysis tool nor a tool to make simulations or reconstruction.

Installation

In this section we describe the installation process in order to get a working copy of sedeb.

Pre-Requisites

Before to start, you will need,:

- Qt4 libraries and development files
- Aerie
- Root compiled with Qt support
- The sedeb sources

The Qt4 Libraries

Depending on your distribution but usually there is a package named:

- Debian and/or Ubuntu: libqt4 and libqt4-dev
- Centos: libqt4 and libqt4-devel

Use your favorite package manager to install it.
But be sure to have installed this libraries before go further on the installation process or sedeb wouldn't compile.

The Aerie installation

There is a guide about how to install aerie step by step
http://private.hawc-observatory.org/hawc.umd.edu/internal/doc.php?id=2219
Please refer to it to get aerie working,

Also you can install aerie and all the packages needed (except Qt4) using ape:
http://private.hawc-observatory.org/wiki/index.php/OS_Requirements

Root compiled with Qt4 support

At the moment sedeb uses libGQt provieded by root, so the dependencies generated by the “histogramming” widgets inside sedeb needs that root is compiled with Qt4; it’s seen that if you have installed previously libQt4 (with it’s development files) the Root compilation with Qt4 will be done with no special treatment if your are using ape.
To check if you have root compiled with Qt4, look at the Root libraries directory if you have: libGQt.so and libQtRoot.so if so you have it.
It’s strongly recommended the use of ape to install the aerie and all the external packages.

Getting sedeb

sedeb is stored in the sandbox area of the personal svn projects for the HAWC Collaboration, you can access it at:

http://private.hawc-observatory.org/svn/hawc/sandbox/cederik/trunk/sedeb

If you want to get the daily sedeb snapshot, perform the next command line into a desired directory:

```
svn co http://private.hawc-observatory.org/svn/hawc/sandbox/cederik/trunk/sedeb sedeb
```

This will creates and populates the directory sedeb on your current directory.
If there is an update on the remote repository you only need to reapply the previous command.

To see how to use svn on HAWC please refer to:
http://private.hawc-observatory.org/wiki/index.php/HAWC_Software_Version_Control

It's not recommended to modify the svn copy (unless you want to make changes to the code and distribute them to the whole collaboration); after downloading make a working copy elsewhere in your system.

Compiling the sedeb

“hard way”

The sedeb compilation process it’s straightforward, just follow the next steps as follows:

1. Set your HAWC environment variables. This can be done:
   go to your ape directory, and run: $ eval `./ape sh` or $ eval `./ape csh` depending the shell you use.
2. Go to the sedeb root directory.
3. Load the ENVARIABLES file: $. ENVARIABLES or $ source ENVARIABLES
4. Run qmake
5. Run make
6. Wait until finish

aerie cmake compatible

This part was tested partially, but the process below works as described, please refer any problem to the author

1. Make a copy from your current svn copy to your aerie source directory (a symlik should work)
2. Proceed as usual when you are compiling aerie from svn
    a. mkdir build && cd build
    b. cmake PATH_FOR_AERIE_SRC -DFLAGS needed (refer to aerie doc)
    c. make sedeb-gui

General sedeb overview

sedeb gui is organized as follows:

- A main window:
    - Menu area:
        - File:
            - Open Files
            - Clear Data
            - Save
            - Quit
        - Edit:
            - Cuts
            - Preferences
        - Tools:
            - Converting tools
        - Help
    - Navigate panel
    - Tab Pages:
        - General Event Info tab-page
        - Detailed Event Info tab-page
        - Simple Histograms tab-page
        - Simple Graph tab-page

Using the sedeb

If you are reading this, it means that the compile process was fine or already have a sedeb working copy. (At any moment you need to have the aerie environment variables set-up, otherwise unexpected issues will appear.)

To start using sedeb, type at the command line:

$ ./sedeb

The config file for first run: (SEDEBConfig.xml)

If you are running for the first time the sedeb, you will be prompted for a config file named SEDEBConfig.xml, unless you attend it, the message will continuously appear. This messages ensures that you have a configuration file to be used every time you run the program, and avoid the user to specify path to aerie configuration files (as detector or fit files)

Do not worry about fill a lot of fields, if you have already the environmental variables for aerie set, the configuration dialog will take the default values, so you need to modify a few lines. See Fig. PREFERENCES 001

Go to:

Edit -> Preferences
(make the changes you will need to do.) After that:
>Save
>Close

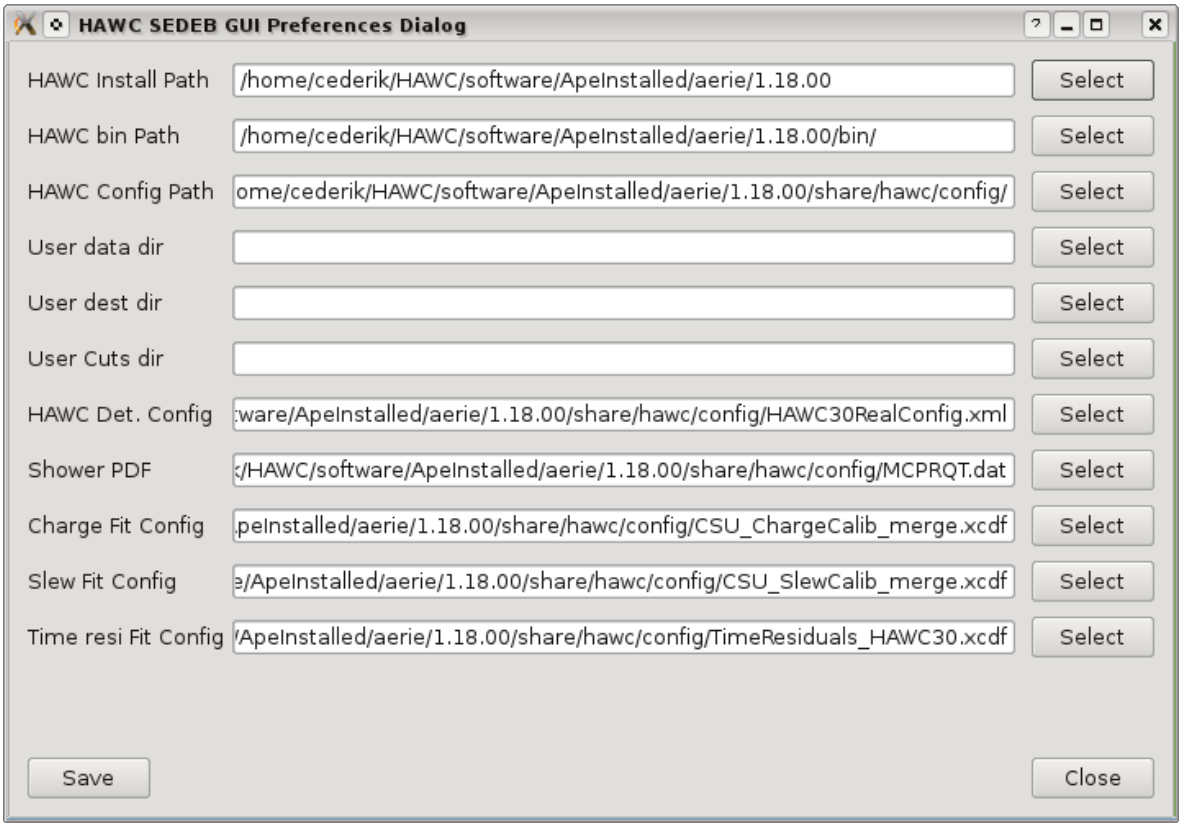


Fig. PREFERENCES 001

It is important to notice if you change the aerie path after you set the sedeb configuration file unexpected issues will occur.

Preferences dialog a detailed view:
Here it's explained every field of the Preferences dialog nevertheless it's self explained,for every field mentioned please see Fig. PREFERENCES 001.

Managing Files

Working with existing files

In this section we will explain the files that sedeb can load and how to do it.

The sedeb, at the moment, works with two type of root files:

1. root files obtained from the triggered files converted with xcdf-root
   (usually will not display information inside detector canvas)
2. root files obtained from the offline-reconstructor
   (it will display information inside detector canvas)

Loading files

MENU: FILE->OPEN ROOT->{OFFLINE/VAMORE RECO, TRIG (XCDF-ROOT)}

To load a file go to main menu:

FILE->OPEN ROOT-> { OFFLINE / VAMOS RECO , TRIG (XCDF-ROOT) }

At this point you have two options:

1. OFFLINE / VAMOS RECO: Will load an root file generated with offline-reconstruction or vamos-reader-reconstructor

2. TRIG (XCDF-ROOT): Will load a root file generated with xcdf-root file converter from a “trig” xcdf file as source

Use the option that suits you. see Fig. OPEN ROOT FILE 001

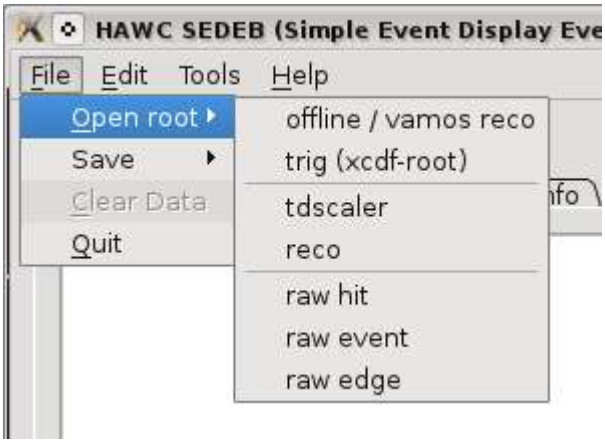


Fig. OPEN ROOT FILE 001

No matter what file type you had choose, a “file dialog” will prompt you to select the desired file(s) to load, at this point you can select one or more files at once. NOTE: Select only the same type of files, do not mix it.

If you want to **add more files** (from the same type) to the existing collection of files already loaded, just repeat the FILE->OPEN ROOT (but alway keeping the same file type) as many times as you need.

Saving

At the moment sedeb can save the images of the current event on detector canvas in a Portable Network Graphics format (PNG) and/or the histograms (1D and 2D); the saving process can be performed in two ways:

1. Menu->Save->{All images, histograms}
   This will save all the images that can be generated on the current sesion.
2. In every element, that allows saving image, click on the SAVE button
   This will save only the associated element with it’s corresponding SAVE button. It means that the General Event Info has it’s own SAVE button that only will save the current canvas detector view; the 1D histogram has it’s own SAVE button that will save the current view (no matter if it is empty), as well as the 2D Histogram

At the moment the default file names are:

- CurrentDetectorPlot.png (for the current detector view)
- 1DHistogram.eps (for the current 1D histogram)
- 2DHistogram.eps (for the current 2D histogram)

Note: on next sedeb releases this can be entered by user-

Clearing the current session:

MENU: FILE->CLEAR DATA
If you decide load a different file type, you must clear all the current data and load the desired file type as previously described, it will prevent program crashes or unpredicted behavior; the clearing process can be done as follows (from main menu), see Fig. CLEAR_DATA001:

FILE->CLEAR DATA
After that, you can load your files, repeating the loading process described above.

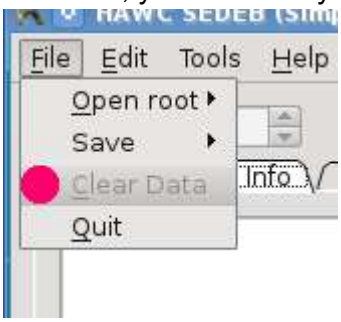


Fig. CLEAR DATA 001

Navigating through events:

sedeb has a simple way to navigate (selecting an event arbitrarily) through the event collection contained in a file or file collection (no matter if the file collection have just one file and if the file contain just one event); after loaded the file(s) at the detector canvas you will see the first event loaded and the the file name will be added to the FILES SELECTED area at the GENERAL EVENT INFO tab, in this way you have notice on what files are you working on. No matter if you change the tab at the main window, you always will have a view of the "NAVIGATION SPIN BOX" Fig. NAV001

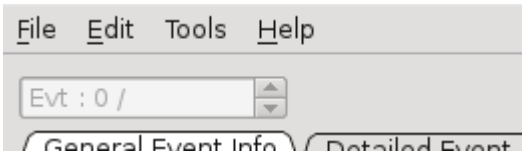


Fig. NAV001. The event navigator

At the event navigator Fig. NAV001 you can use the up/down arrows, clicking on it, this will increment or decrement by 1 the event number and automatically will update the information on the displayed event (the leftmost number is the current event on screen, the rightmost number is the total number of events loaded on the current session), you can enter the number directly from your keyboard, just click on the NAVIGATOR SPIN BOX and type the desired event. If you have a mouse or "touch pad" with scroll functionality it will works also.

The Tab-Pages

The Tab-Pages provides a tab bar and a "page area" that is used to display pages related to each tab; sedeb contains the next tabs:

- **General Event Info**: This tab contains the main detector canvas constructed by the aerie detector class implementation on sedeb, on this tab you can see, if the file format allow it, which tanks and pmts are related with the event selected by the navigation spin box, you can turn {on, off} the tanks, pmts, and/or the core {gauss, guess}(if apply), displays an area with the files involved in the current session. See Fig. GENERAL EVENT INFO 001

- **Data Detailed Event Info**: It is a text based area where all the information, related to the event extracted from the file type you choose, is dumped. Can be selected and copied to the system clipboard. For every new event selected the text will be replaced. Also have a “save image” button it will save the current detector canvas image on a png file. See Fig. DETAILED EVENT INFO 001

- **Histograms**: This window contains two type of histograms managed by the root libraries. 1D (TH1F) and 2D (TH2F); depending on the file type you choose the variables will be setup on the combo box related to each object. (we will discuss it later) See Fig. EVENT HISTOGRAMS 001

- **Simple Plot**: This part is under development and it’s was mainly intended to make x-y graphics, using just a selected group of variables that suits that purpose. (we will discuss it later)

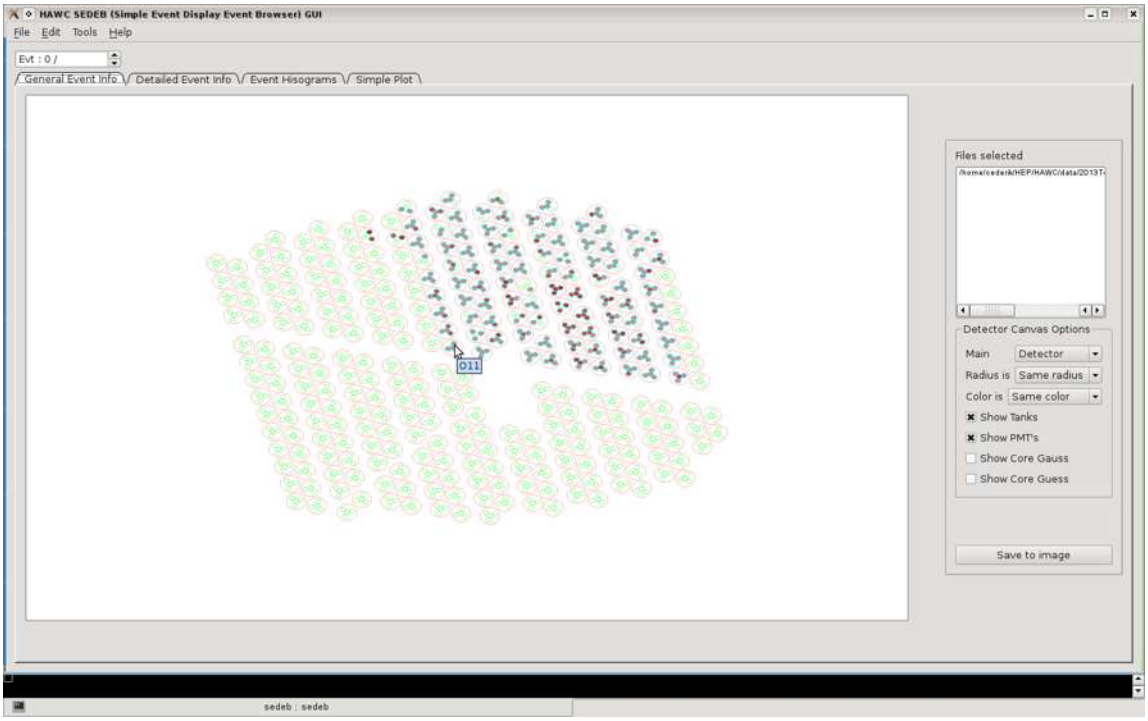


Fig. GENERAL EVENT INFO 001

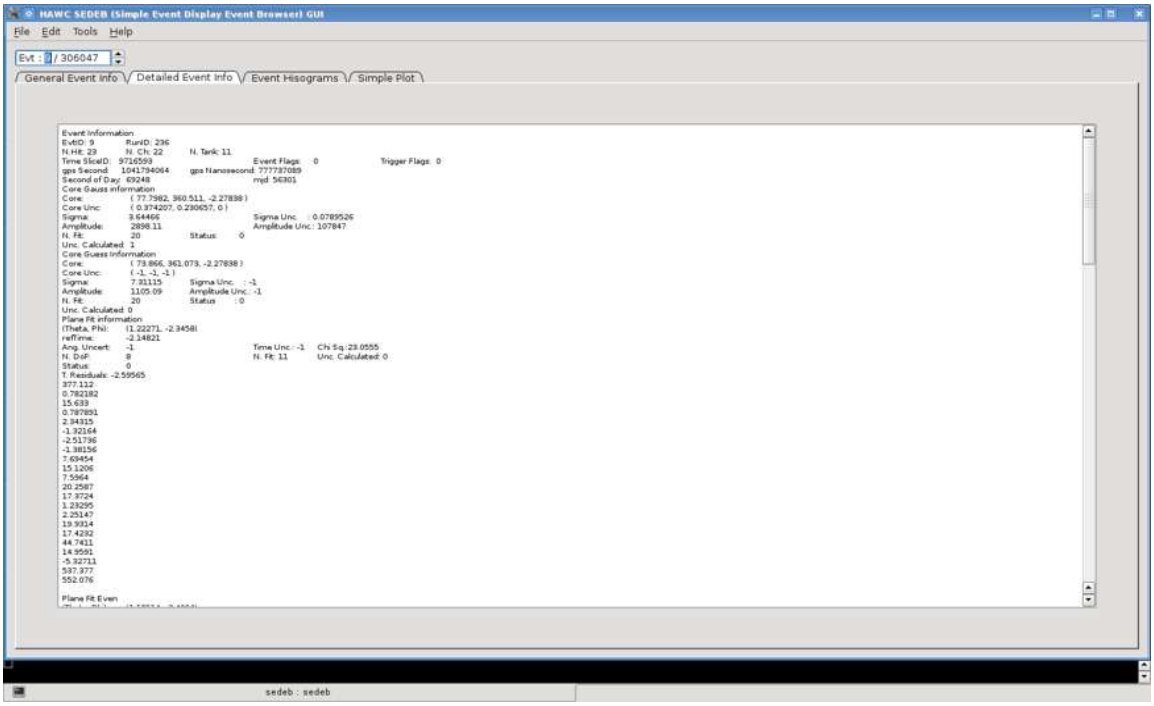


Fig. DETAILED EVENT INFO 001

Event Histograms

After a file is loaded you can access to the Event Histograms Tab page, in this page you will find a set of two histograms types: 1D Histogram and 2D Histogram; constructed with the TH1F and TH2F Root classes; the 1D Histogram is located at the leftmost part of this Tab, and the 2D histogram is located at the rightmost part of the tab.
On both cases you only need to specify a variable (1D histogram), or variables (2D histogram) selecting them at the bottom combo box on each histogram; automatically will update after selection changes. Note: if you are working with big files (GB). some histograms, depending on the variables selected, will take more time (tens of seconds or more).
You can select the variable to plot using the mouse or through the arrow keys (up/down)
Also you can set the Histogram title clicking on the title text box and entering the desired text to be shown. See Fig. EVENT HISTOGRAM 001

Saving the histograms to a file

Below each histogram is a SAVE button, this will save the current histogram view into an eps file format.

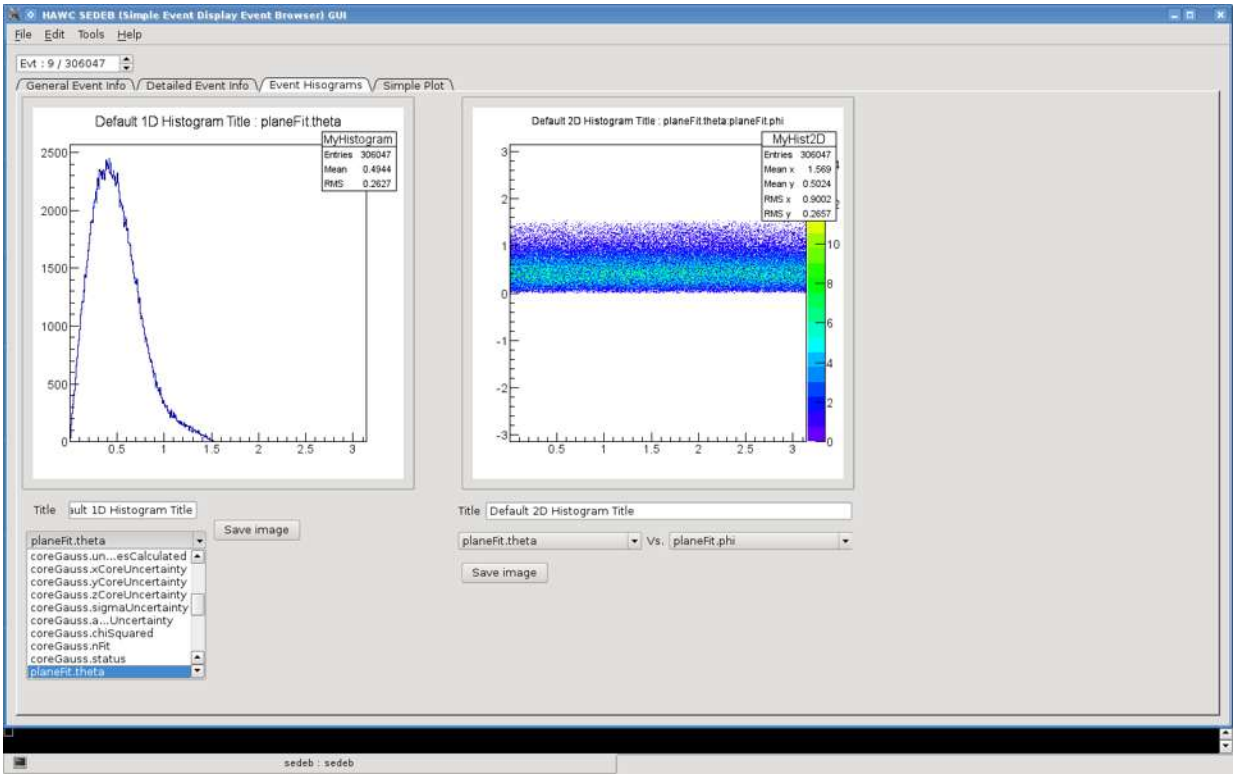


Fig. EVENT HISTOGRAM 001

Managing Cuts

MENU: EDIT->CUTS

A cut is a set of conditions imposed to parameters, to make a list of events with a fine selection on which only the events that satisfied that conditions are used.
It's important to notice that, in sedeb, after you apply a cut, the statistics, histograms and graphics will operate with that list until you apply another cut.

The cuts dialog provides the user a tool that can be used to select a list of events with a set of properties; can experiment with different values on cuts until be satisfied.

The main cuts dialog it's splitted into three parts see Fig. CUTS 001:

1. **Saved Cuts**: It's an area that shows and allow the user to select (with double click on them), cuts stored on xml files; this are user saved cuts. When mouse pointer is over a cut it will display a description on that cut

2. **Cuts editor**: Allows the user to select the variables, comparison operators, enter the values as well the logic operators and put into a text area, can be tested and, if the user wants, save it for a future use.

3. **List of events**: It's an interactive area that shows the event list and only contains the events that passed the cuts, with double click on desired event will update all the General Event Info and the Detailed Event Info. It is important to note that the statistics and histograming and plots runs over the elements of that list.

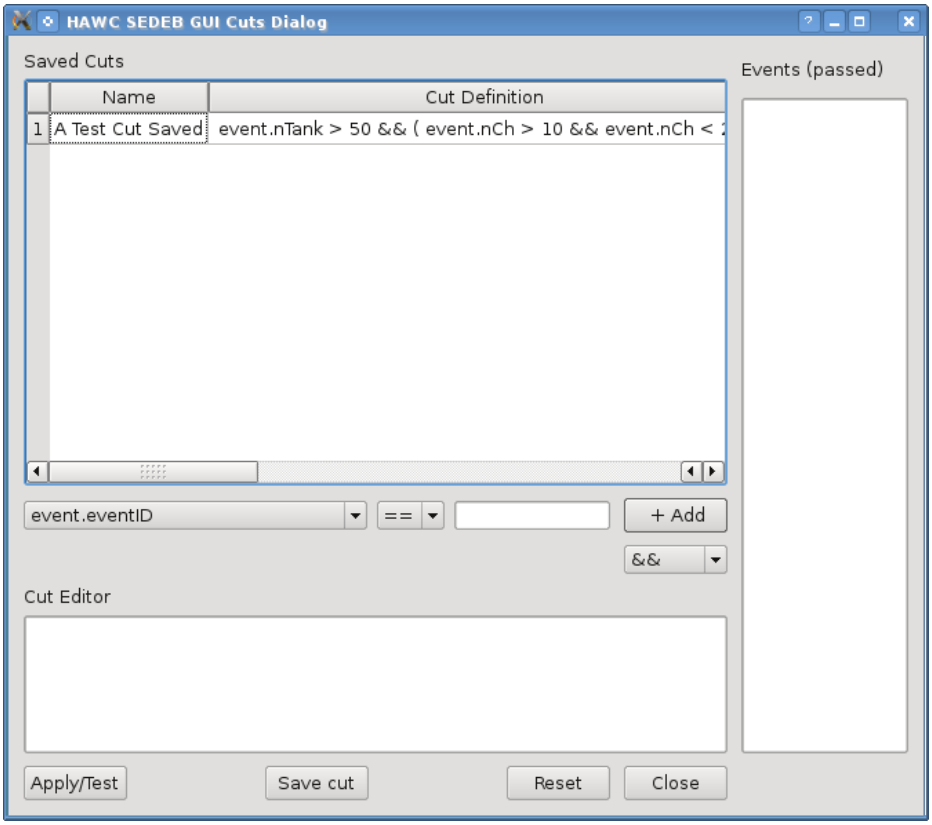


Fig. CUTS 001

The cut process can be performed as many times as required:

1. just select the variables,
2. set the arithmetic operator
3. set requested value and,
4. if required, the logic operator.

Then apply the cut and wait until it finish; refer to event list and see what events passed the cuts.

If you need to change a value or an operator in your cut, you can do it directly at the cut editor text edit area, then apply the cut again.

If you need to make an entirely new cut and discard all what the Cut editor area have, just press the RESET button. Note: this will not apply a new cut, just erase the window contents, so if you provided a cut and apply it, it will persist until you apply a new one.

Cuts saving

If you want save the cuts for a future use, just click on the SAVE CUT button, a dialog will appear and you only need to fill the fields, of course it is not mandatory but recommended (only the cut name is mandatory, until you fill the field, the SAVE button will be enabled), because if you are sharing cuts you may want to know more about : What the cut perform? How it does? and Who construct it? See Fig. SAVE CUT 001

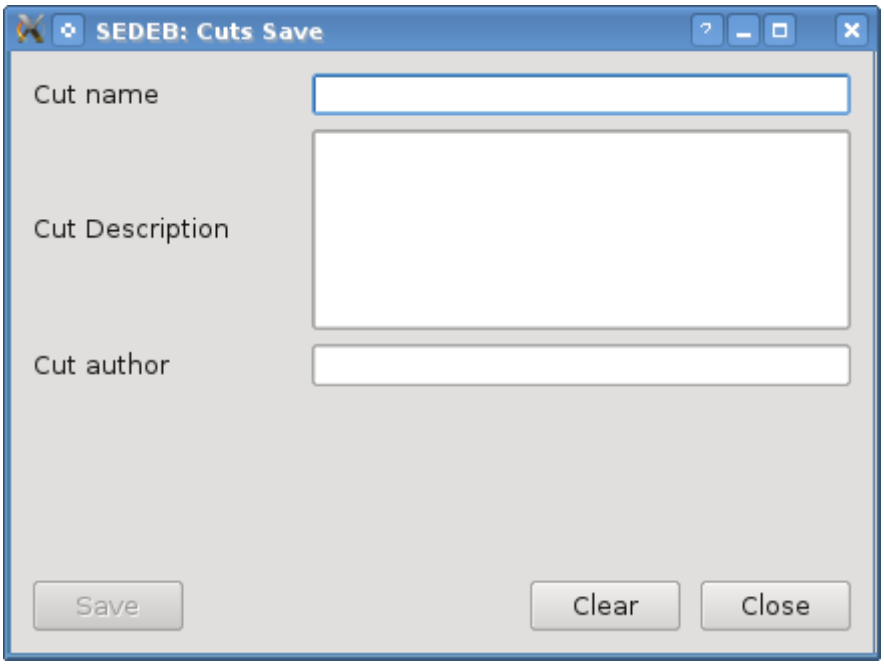


Fig. CUTS SAVE 001

Note: at this point when press the SAVE button, the dialog will prompt you for a location to save the file that contain the cut, you can select any location you want. By default you will be located on the path specified on SEDEBConfig.xml; but if you select a different path to save your cuts you will be unable to retrieve it, until the file is on the path defined on the SEDEBConfig.xml.

Converting files

MENU: TOOLS->FORMAT CONVERTER

This is a simple graphical front-end to the aerie file format converter and reconstruction tools, it helps the user convert xcdf files to a root format or reconstruct events stored in the xcdf files; selecting desired tool for it. Also will produce xcdf files too. (to be done).

This dialog have five areas:

1. **File selector** (Source and Target): The source File can be a set of files (same type files are expected) or only one file. While the Target file is just one filename;

2. **Conversion tool box**: Contains a tool set, the user can select the tool to be used to perform the conversion or reconstruction, depending on he/she needs;

3. **Parameter selection**: The set of parameters related to a tool; by default all the options are disabled until the tool is selected then the associated parameters are enabled.

4. **Output Log window**: A text area that inform the used about what is going passing all the result from standard input/output and standard error generated by the current process;

5. **Control buttons**: A set of buttons that allow the user to control the process and dialog.

See Fig. FILE FORMAT CONVERTER 001

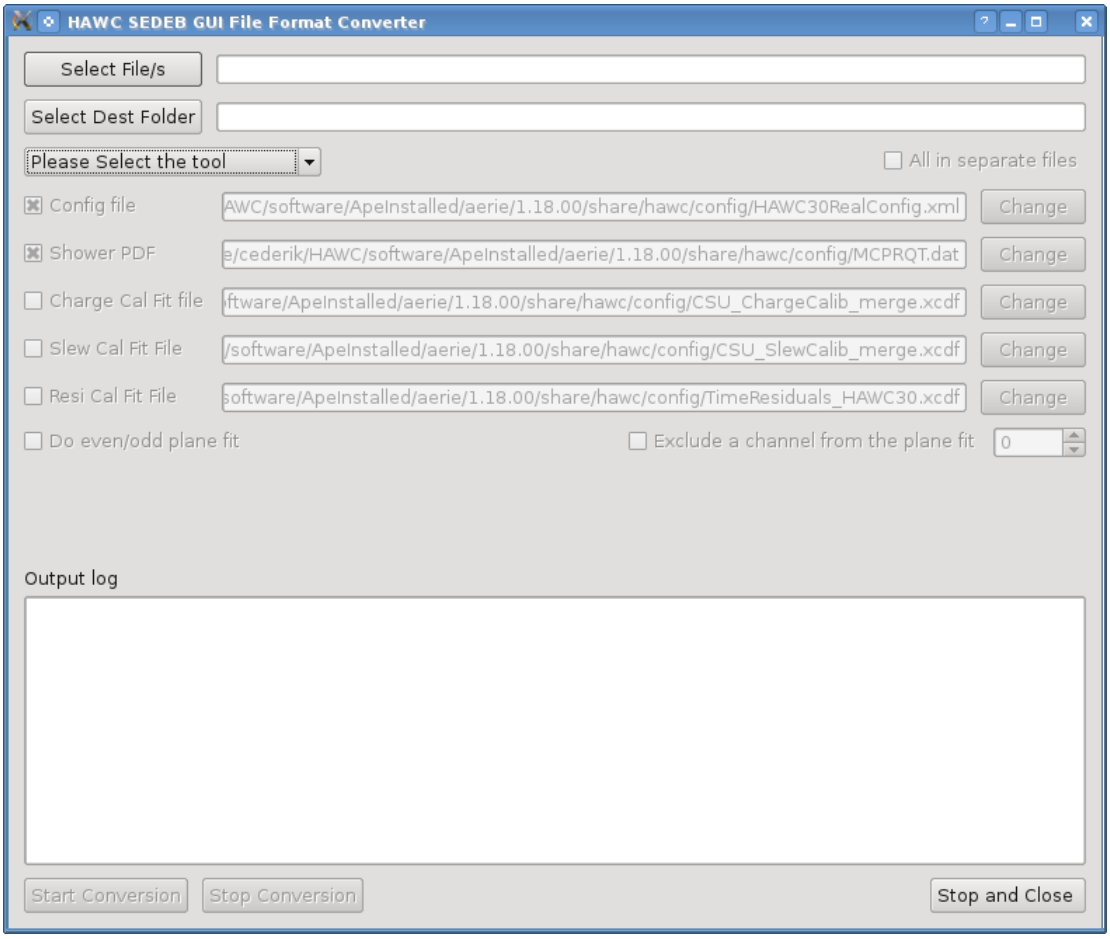


Fig. FILE FORMAT CONVERTER 001

The file format converter can perform over files:

- **one-to-one process**: you can select a single file (through the select file dialog) as a source file, then in the target field you can specify the root file name (this process is not destructive, your source file will be safe)

- **many-to-one**: as previous, the main difference it's that you can select as many files as you want and then generate o single root file;

- **many-to-many:** Selecting multiple files as source and covert or generate reconstructed it, resulting as many target files as source (to be implemented, coming soon!)

The process it’s quite simple:

1. Select your source(s) files;
2. Select your target file(s);
3. Select the tool to manipulate the files provided above;
4. Select desired parameters (some of them are set by default, unless you know what are

you doing keep it as it comes, this values are set by the environment variables at the beginning)

5. Press START CONVERSION button, and see what the Window Log prints
    a. If you want only stop the running process press STOP CONVERSION button
    b. If you want stop the running process and close the conversion dialog press STOP AND CLOSE button (the safest way to close this window)

This dialog and a related process can stay running at the same time you continue your work with sedeb, it means that while the conversion or reconstruction process is done you can do whatever you want in sedeb, even load a file that is in the middle of it generating process (this is not recommended). A progress bar will indicate if the process is still in progress with a moving colored bar that appear immediately the START button is pressed, when the process is done the progress bar will disappear. The status of a progress is always monitored on the Log window area.

Help

This Area will provide a user guide and information related to the version of the sedeb.

Objectives: Develop a tool to visualize and navigate through a collection of files that contains event data from HAWC Experiment

Features:

Can open and display the information related to the Root event file (raw -> xcdf ->root)

Can append to the current event collection a new Root event file (raw-> xcdf->root)

Uses the Root mechanism to look and search events with the cuts implementation

Display 1-D and 2-D histograms from the user selected Branches

Display a simple event canvas and show user selected observable (xPMT, yPMT, etc) directly from branch names

APÉNDICE D

DOCUMENTACIÓN DE CLASES DE SEDEB

Reference Manual

Generated by Doxygen 1.8.1.2

Thu Aug 22 2013 13:52:20

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 Ui Namespace Reference

4.1.1 Detailed Description

Here we will include the definitions for External Windows and Dialogs About Plotting

Chapter 5

Class Documentation

5.1 AboutDialog Class Reference

Public Member Functions

- AboutDialog (QWidget *parent=0)
- ~AboutDialog ()

5.1.1 Constructor & Destructor Documentation

5.1.1.1 AboutDialog::AboutDialog (QWidget * *parent* = 0) [explicit]

The AboutDialog default constructor

5.1.1.2 AboutDialog::~AboutDialog ( )

The AboutDialog default destructor

The documentation for this class was generated from the following files:

- include/sedeb/aboutdialog.h
- src/aboutdialog.cpp

5.2 compactness40 Class Reference

`#include <compactness40.h>`

Public Member Functions

- compactness40 ()

Public Attributes

- double **radius**
- double **compactness**
- double **maxPE**
- unsigned long long **nFit**
- unsigned long long **status**

5.2.1 Detailed Description

The compactness40 class

5.2.2 Constructor & Destructor Documentation

5.2.2.1 compactness40::compactness40 ( )

The compactness40 constructor

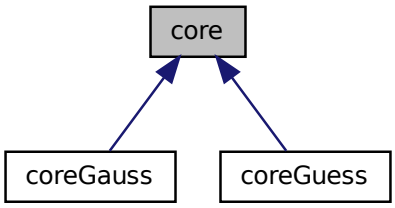
The documentation for this class was generated from the following files:

- include/sedeb/compactness40.h
- src/RootFileFormat.cpp

5.3 core Class Reference

`#include <core.h>`

Inheritance diagram for core:



Public Attributes

- double **xCore**
- double **yCore**
- double **zCore**
- double **sigma**
- double **amplitude**
- double **xCoreu**
- double **yCoreu**
- double **zCoreu**
- double **sigmau**
- double **amplitudeu**
- double **chiSq**
- unsigned long long **nFit**
- unsigned long long **uCalculated**
- unsigned long long **status**

5.3.1 Detailed Description

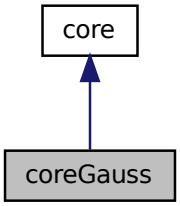
The coreGauss class

The documentation for this class was generated from the following files:

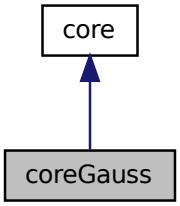
- include/sedeb/core.h
- src/RootFileFormat.cpp

5.4 coreGauss Class Reference

Inheritance diagram for coreGauss:



Collaboration diagram for coreGauss:



Public Member Functions

- coreGauss ()

Additional Inherited Members

5.4.1 Constructor & Destructor Documentation

5.4.1.1 coreGauss::coreGauss ( )

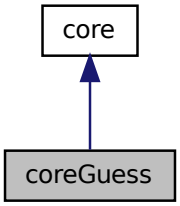
The coreGauss constructor

The documentation for this class was generated from the following files:

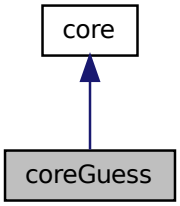
- include/sedeb/core.h
- src/RootFileFormat.cpp

5.5 coreGuess Class Reference

Inheritance diagram for coreGuess:



Collaboration diagram for coreGuess:



Public Member Functions

- coreGuess ()

Additional Inherited Members

5.5.1 Constructor & Destructor Documentation

5.5.1.1 coreGuess::coreGuess ( )

The coreGuess constructor

The documentation for this class was generated from the following files:

- include/sedeb/core.h
- src/RootFileFormat.cpp

5.6 CutsDialog Class Reference

Signals

- void **EventSelected** (int)

Public Member Functions

- CutsDialog (QWidget *parent=0, RootFileFormat *RootFF=0)
- ~CutsDialog ()
- void GetCutedEvents ()
- void **SetSource** (RootFileFormat *RFFSource)
- void InitializeLoadSavedCuts (void)
- void LoadSavedCuts (void)

5.6.1 Constructor & Destructor Documentation

5.6.1.1 CutsDialog::CutsDialog (QWidget * *parent* = 0, RootFileFormat * *RootFF* = 0) [explicit]

The CutsDialog default constructor

5.6.1.2 CutsDialog::~CutsDialog ( )

The CutsDialog default destructor

5.6.2 Member Function Documentation

5.6.2.1 void CutsDialog::GetCutedEvents ( )

This method allows the user pass the cuts string from the GUI,

Parameters

RoofFileFormat	class pointer to the collection

Please OVERRIDE with a config dialog

5.6.2.2 void CutsDialog::InitializeLoadSavedCuts (void)

This method will initialize the table widget in order to be filled with the data from the xml cut files

5.6.2.3 void CutsDialog::LoadSavedCuts (void)

Will initialize the widget load cuts

This method setup and fills the tablewidged with the contents of every xml file from the cuts directory, at the moment we will look for ∗.cut files at the app dir. the file must be at xml format This part shoul be modified to resize the row heigh

The documentation for this class was generated from the following files:

- include/sedeb/cutsdialog.h
- src/cutsdialog.cpp

5.7 cutssave Class Reference

Public Member Functions

- **cutssave** (QWidget ∗parent=0)
- void **SetCutDataString** (QString CutDataString)

The documentation for this class was generated from the following files:

- include/sedeb/cutssave.h
- src/cutssave.cpp

5.8 detectorcanvas Class Reference

Public Member Functions

- detectorcanvas ()
- ∼detectorcanvas ()
- void SetCanvas (MyQGraphicsView ∗CanvasDetector)
- void SetCanvasDimensions (double Rw, double Rh, double Cw, double Ch, double resolution)
- void SetCanvasGrid (double w, double h)
- void **DrawTanks** (bool DrawIt)
- void **DrawPMTS** (bool DrawIt)
- void DrawData (Event ∗EventToDraw)
- void DrawDetector ()
- void SaveSnapshot ()
- void SetTankRadius (double TankR)
- void SetPMTRadius (double PMTR)
- void SetDetectorBounds (double xMinB, double xMaxB, double yMinB, double yMaxB)
- void DrawCoreGauss (coreGauss ∗CoreGauss2Draw)
- void **DrawCoreGuess** (coreGuess ∗CoreGuess2Draw)

5.8.1 Constructor & Destructor Documentation

5.8.1.1 detectorcanvas::detectorcanvas ( )

detectorcanvas default constructor The Main Canvas designed for Plot

5.8.1.2 detectorcanvas::~detectorcanvas ( )

detectorcanvas default destructor The Main Canvas designed for Plot

5.8.2 Member Function Documentation

5.8.2.1 void detectorcanvas::DrawCoreGauss (coreGauss * *CoreGauss2Draw*)

This methods are experimental

This method is experimental, so please be careful will draw the associated coreGauss and coreGuess to an event

5.8.2.2 void detectorcanvas::DrawData (Event * *EventToDraw*)

This method draws a graphical representation from a collection of data into the canvas

5.8.2.3 void detectorcanvas::DrawDetector ( )

This method will draw as a representation the data sent

This method draw the current detector using the geometry provided by the aerie framework

5.8.2.4 void detectorcanvas::SaveSnapshot ( )

This method allow to save in a png file needs a lot of implementation and debug

5.8.2.5 void detectorcanvas::SetCanvas (MyQGraphicsView * *CanvasDetector*)

SetCanvas this method set the very basic properties for the canvas detector

Parameters

CanvasDetector	set where to implement the changes and settings

5.8.2.6 void detectorcanvas::SetCanvasDimensions (double *Rw,* double *Rh,* double *Cw,* double *Ch,* double *resolution*)

SetCanvasDimensions Sets the canvas dimensions

Parameters

Rh	The real detector Geometry Heigh
Rw	The real detector Geometry Width
Ch	The real detector Geometry Heigh
Cw	The real detector Geometry Width

5.8.2.7 void detectorcanvas::SetCanvasGrid (double *h,* double *w*)

This method builds the very basic grid into the canvas Needs a lot of work on transformations Draws the vertical grid mini lines

we should compute the label value outside

Draws the horizontal grid mini lines

we should compute the label value outside

5.8.2.8 void detectorcanvas::SetDetectorBounds (double *xMinB,* double *xMaxB,* double *yMinB,* double *yMaxB*)

This method set the Detector initial bound, it will be recalculated every draw event method call

5.8.2.9 void detectorcanvas::SetPMTRadius (double *PMTR*)

This method set the PMT radius to be drawn at canvas

5.8.2.10 void detectorcanvas::SetTankRadius (double *TankR*)

This method set the Tank radius to be drawn at canvas

The documentation for this class was generated from the following files:

- include/sedeb/detectorcanvas.h
- src/detectorcanvas.cpp

5.9 Event Class Reference

The class for the Event from the Root File Format Generated by reconstruction AERIE tools.

`#include <Event.h>`

Public Member Functions

- Event ()

Public Attributes

- unsigned long long **eventID**
- unsigned long long **runID**
- unsigned long long **timeSliceID**
- unsigned long long **gpsSecond**
- unsigned long long **gpsNanosecond**
- unsigned long long **mjd**
- unsigned long long **secondOfDay**
- unsigned long long **nHit**
- unsigned long long **nCh**
- unsigned long long **nTank**
- unsigned long long **eventFlags**
- unsigned long long **triggerFlags**
- unsigned long long **gtcFlags**
- long long **laserTStart**
- long long **laserTStop**
- long long **laserLightToTanksStart**
- long long **laserLightToTanksStop**
- unsigned long long * **hitgridId**
- unsigned long long * **hittankId**
- unsigned long long * **hitchannelIsGood**
- unsigned long long * **hitloTOT**
- unsigned long long * **hithiTOT**
- unsigned long long * **hittime01**
- unsigned long long * **hitflags**

- unsigned long long * **hittriggerFlags**
- long long * **hitrawTime**
- double **sumPE**
- double * **hitxPMT**
- double * **hityPMT**
- double * **hitzPMT**
- double * **hittime**
- double * **hitcharge**
- double * **hitloTOTCharge**
- double * **hithiTOTCharge**

5.9.1 Detailed Description

The class for the Event from the Root File Format Generated by reconstruction AERIE tools.

Author

Cederik L. De León A.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Event::Event ( )

The Event Class default constructor

The documentation for this class was generated from the following files:

- include/sedeb/Event.h
- src/RootFileFormat.cpp

5.10 FFDialog Class Reference

Public Slots

- void RunProc (QString Command, QStringList Arguments)
- void SendToStdOut ()
- void KillProc ()

Public Member Functions

- **FFDialog** (QWidget *parent=0)
- void LoadValues ()
- QStringList LoadFFConfPathDialog ()
- void DisabledAllFields ()
- void LoadProgramList ()

5.10.1 Member Function Documentation

5.10.1.1 void FFDialog::DisabledAllFields ( )

This method disabled all the field in the File Format Converter Dialog

5.10.1.2 void FFDialog::KillProc ( ) [slot]

This will kill the process if there is one we must add an stdout error return value

5.10.1.3 QStringList FFDialog::LoadFFConfPathDialog (void)

This method allows to load a new File (user selected) and use it in a current session, the values of the config fields will be overwrite every time that the user opens the file converter dialog, so if need to preserve the values, you need to modify the main preferemnces in PREFERENCES dialog and save it.

5.10.1.4 void FFDialog::LoadProgramList ( )

This method fill the FFSelectComboBox with a list of bin programs used to manipulate files

5.10.1.5 void FFDialog::LoadValues ( )

This method load the values from the preferences file into the GUI line edits

5.10.1.6 void FFDialog::RunProc (QString *Command,* QStringList *Arguments*) [slot]

This method will run the external process and connect its output to the stdsouttextedit widget

5.10.1.7 void FFDialog::SendToStdOut ( ) [slot]

This method send the result of the terminal execution to a text box

The documentation for this class was generated from the following files:

- include/sedeb/ffdialog.h
- src/ffdialog.cpp

5.11 histogram Class Reference

Public Slots

- void **SaveHistogram** ()

Public Member Functions

- histogram (QWidget *parent=0)
- void SetName (string)
- void SetTitle (string)
- void SetVarToPlot (string TheVarToPlot)
- void FillHist (TChain *)

5.11.1 Constructor & Destructor Documentation

5.11.1.1 histogram::histogram (QWidget * *parent* = 0)

The Default Histogram Constructor

5.11.2 Member Function Documentation

5.11.2.1 void histogram::FillHist (TChain * *MChain*)

The FillHistogram Method

Parameters

MChain	tells where to get the information

5.11.2.2 void histogram::SetName (string *TheHistoName*)

The SetName Method

Parameters

TheHistoName	specifies the name for the histogram

5.11.2.3 void histogram::SetTitle (string *TheHistoTitle*)

The SetTitle Method

Parameters

TheHistoTitle	speficies the title to show in the histogram

5.11.2.4 void histogram::SetVarToPlot (string *TheVarToPlot*)

The SetVarToPlot Method

Parameters

TheVarToPlot	it's a string containing the branch to plot

The documentation for this class was generated from the following files:

- include/sedeb/histogram.h
- src/histogram.cpp

5.12 histogram2d Class Reference

Public Slots

- void **SaveHistogram** ()

Public Member Functions

- **histogram2d** (QWidget *parent=0)
- void SetName (string)
- void SetTitle (string)
- void SetVarToPlot (string TheVarToPlotA, string TheVarToPlotB)
- void FillHist (TChain *)

5.12.1 Member Function Documentation

5.12.1.1 void histogram2d::FillHist (TChain * *MChain*)

The FillHistogram Method (For the 2D Histogram)

Parameters

MChain	tells where to get the information

Must have the option user decides

5.12.1.2 void histogram2d::SetName (string *TheHistoName*)

The SetName Method

Parameters

TheHistoName	specifies the name for the histogram

5.12.1.3 void histogram2d::SetTitle (string *TheHistoTitle*)

The SetTitle Method

Parameters

TheHistoTitle	speficies the title to show in the histogram

5.12.1.4 void histogram2d::SetVarToPlot (string *TheVarToPlotA*, string *TheVarToPlotB*)

The SetVarToPlot Method

Parameters

TheVarToPlot	it's a string containing the branch to plot

Empties the previous values to reuse

Set the values

The documentation for this class was generated from the following files:

- include/sedeb/histogram2d.h
- src/histogram2d.cpp

5.13 lhFit Class Reference

`#include <lhFit.h>`

Public Member Functions

- lhFit ()
- ~lhFit ()

Public Attributes

- double **theta**
- double **phi**
- double **refTime**
- double **angularu**
- double **timeu**
- double **chiSq**
- unsigned long long **Ndof**
- unsigned long long **nFit**
- unsigned long long **ucalculated**
- unsigned long long **status**
- double * **tResi**

5.13.1 Detailed Description

The lhFit Class

5.13.2 Constructor & Destructor Documentation

5.13.2.1 lhFit::lhFit ( )

The lhFit constructor

5.13.2.2 lhFit::~lhFit ( )

The lhFit destructor

The documentation for this class was generated from the following files:

- include/sedeb/lhFit.h
- src/RootFileFormat.cpp

5.14 MainWindow Class Reference

Public Member Functions

- MainWindow (QWidget *parent=0)
- ~MainWindow ()
- void RetrieveData (void)
- void DataFill (void)
- void VariablesSet (void)
- void TrigVariablesSet (void)
- void SetFileType (string FileTypeL)
- string GetFileType (void)

5.14.1 Constructor & Destructor Documentation

5.14.1.1 MainWindow::MainWindow (QWidget * *parent* = 0) [explicit]

The next lines setup the main canvas detector

5.14.1.2 MainWindow::~MainWindow ( )

The default MainWindow Destructor

5.14.2 Member Function Documentation

5.14.2.1 void MainWindow::DataFill (void)

Will fill all data into memory, example user app

This function should be called into an Event Update Take care

5.14.2.2 string MainWindow::GetFileType (void)

This function ask for the file type loaded

5.14.2.3 void MainWindow::RetrieveData (void)

This function retrieves all data from Root Branches and assign to it's respective class This function sets the name of the file to be readed and set the QStringList to be used by the GUI in order to fill it's contents

This function sets the name of the file to be readed and set the QStringList to be used by the GUI in order to fill it's contents

5.14.2.4 void MainWindow::SetFileType (string *FileTypeL*)

This function set the file type in order to access the correct functions

5.14.2.5 void MainWindow::TrigVariablesSet (void)

Will set all the variable set to be used by the GUI

This function loads a trig file generated with xcdf-root converter

5.14.2.6 void MainWindow::VariablesSet (void)

Will fill all data into Qt, example user app

This function is just for the vamors-reader-reconstruct root file generated in a future implementation we must pass as parameter the file tipe to setup all the root file generated files and manage them

The documentation for this class was generated from the following files:

- include/sedeb/mainwindow.h
- src/mainwindow.cpp

5.15 MyQGraphicsView Class Reference

Public Slots

- void **scalingTime** (qreal x)
- void **animFinished** ()

Public Member Functions

- **MyQGraphicsView** (QWidget *parent=0)
- void **wheelEvent** (QWheelEvent *event)

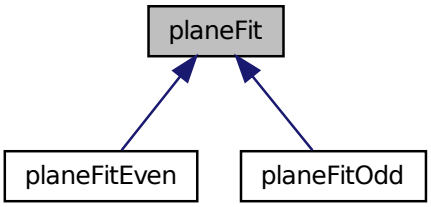
The documentation for this class was generated from the following files:

- include/sedeb/detectorcanvas.h
- src/detectorcanvas.cpp

5.16 planeFit Class Reference

`#include <planeFit.h>`

Inheritance diagram for planeFit:



Public Member Functions

- planeFit ()

Public Attributes

- double **theta**
- double **phi**
- double **refTime**
- double **angularu**
- double **timeu**
- double **chiSq**
- unsigned long long **Ndof**
- unsigned long long **nFit**
- unsigned long long **ucalculated**
- unsigned long long **status**
- double * **tResi**

5.16.1 Detailed Description

The planeFit Class

5.16.2 Constructor & Destructor Documentation

5.16.2.1 planeFit::planeFit ( )

The planeFit constructor

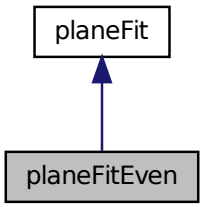
The documentation for this class was generated from the following files:

- include/sedeb/planeFit.h
- src/RootFileFormat.cpp

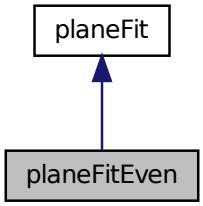
5.17 planeFitEven Class Reference

`#include <planeFit.h>`

Inheritance diagram for planeFitEven:



Collaboration diagram for planeFitEven:



Public Member Functions

- planeFitEven ()

Additional Inherited Members

5.17.1 Detailed Description

The planeFitEven Class

5.17.2 Constructor & Destructor Documentation

5.17.2.1 planeFitEven::planeFitEven ( )

The planeFitEven constructor

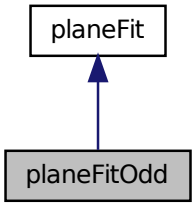
The documentation for this class was generated from the following files:

- include/sedeb/planeFit.h
- src/RootFileFormat.cpp

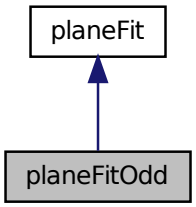
5.18 planeFitOdd Class Reference

`#include <planeFit.h>`

Inheritance diagram for planeFitOdd:



Collaboration diagram for planeFitOdd:



Public Member Functions

- planeFitOdd ()

Additional Inherited Members

5.18.1 Detailed Description

The planeFitOdd Class

5.18.2 Constructor & Destructor Documentation

5.18.2.1 planeFitOdd::planeFitOdd ( )

The planeFitOdd constructor

The documentation for this class was generated from the following files:

- include/sedeb/planeFit.h
- src/RootFileFormat.cpp

5.19 preferences Class Reference

Public Member Functions

- **preferences** (QWidget *parent=0)
- QStringList **GetPreferences** ()
- void **SetPreferences** ()
- void CreateNewPrefFile ()
- QStringList **LoadPrefPathsConfDlg** (bool IWantADir)

Public Attributes

- QString **hawc_install**
- QString **hawc_config**
- QString **hawc_bin**
- QString **hawc_detector**
- QString **hawc_showerPDF**
- QString **hawc_slew**
- QString **hawc_charge**
- QString **hawc_timeres**
- QString **hawc_datadir**
- QString **hawc_destdir**
- QString **hawc_cutsdir**

5.19.1 Member Function Documentation

5.19.1.1 void preferences::CreateNewPrefFile ( )

This method only is used when the default config file isn't present and makes a new one First we need to fill the dialog fields

The documentation for this class was generated from the following files:

- include/sedeb/preferences.h
- src/preferences.cpp

5.20 RootFileFormat Class Reference

a class to retreieve an store the content of the HAWC root file created with the reconstruction AERIE tools

`#include <RootFileFormat.h>`

Public Member Functions

- RootFileFormat (std::string MFileType)
- ~RootFileFormat ()
- Event * GetEvent ()
- coreGuess * GetcoreGuess ()
- coreGauss * GetcoreGauss ()
- planeFit * GetplaneFit ()
- planeFitOdd * GetplaneFitOdd ()
- planeFitEven * GetplaneFitEven ()
- lhFit * GetlhFit ()
- TChain * GetTChain ()
- compactness40 * Getcompactness40 ()
- stdCuts * GetstdCuts ()
- stdCutsOdd * GetstdCutsOdd ()
- stdCutsEven * GetstdCutsEven ()
- RootTrig * GetXCDFBranchs ()
- void Initialize (std::string *, int, int)
- void ToEvent (int)
- int GetMaxEntries (void)
- void NextEvent (void)
- void PreviousEvent (void)
- void GetArraySize (int)
- int SetArraySize (void)
- void SetRecoBranch (void)
- void SetTrigBranch (void)
- TEventList * SetCuts (std::string)
- std::string GetEventInfo (Event *)
- std::string GetcoreGuessInfo (coreGuess *)
- std::string GetcoreGaussInfo (coreGauss *)
- std::string GetplaneFitInfo (planeFit *, int)
- std::string GetplaneFitOddInfo (planeFitOdd *, int)
- std::string GetplaneFitEvenInfo (planeFitEven *, int)
- std::string GetlhFitInfo (lhFit *)
- std::string Getcompactness40Info (compactness40 *)
- std::string GetstdCutsInfo (stdCuts *)
- std::string **GetXCDFTrigInfo** (RootTrig *)

Public Attributes

- TCut * **MyCut**
- std::string **FileType**

5.20.1 Detailed Description

a class to retreieve an store the content of the HAWC root file created with the reconstruction AERIE tools

The RootFileFormat class

Author

Cederik De León.

Date

this function launch the file dialog and load a set of files to be loaded with

5.20.2 Constructor & Destructor Documentation

5.20.2.1 RootFileFormat::RootFileFormat (std::string *MFileType*)

Constructor for the main root File Reader classThe default RootFileFormat Constructor

5.20.2.2 RootFileFormat::~RootFileFormat ( )

The RootFileFormat destructor

5.20.3 Member Function Documentation

5.20.3.1 void RootFileFormat::GetArraySize (int *EventNumber*)

This method get the number of elements in array, related to event.nHit

5.20.3.2 compactness40 * RootFileFormat::Getcompactness40 ( )

The Getcompactess40 method

5.20.3.3 string RootFileFormat::Getcompactness40Info (compactness40 * *compactnessData*)

The Getcompactness40Info method allow us get the information in free format

5.20.3.4 coreGauss * RootFileFormat::GetcoreGauss ( )

The GetcoreGauss method

5.20.3.5 string RootFileFormat::GetcoreGaussInfo (coreGauss * *cgaussData*)

The GetcoreGausstInfo method allow us get the information in free format

5.20.3.6 coreGuess * RootFileFormat::GetcoreGuess ( )

The GetcoreGuess method

5.20.3.7 string RootFileFormat::GetcoreGuessInfo (coreGuess * *cguessData*)

The GetcoreGuesstInfo method allow us get the information in free format

5.20.3.8 Event * RootFileFormat::GetEvent ( )

The GetEvent method

5.20.3.9 string RootFileFormat::GetEventInfo (Event * *eData*)

The GetEventInfo method allow us get the information in free format

5.20.3.10 lhFit * RootFileFormat::GetlhFit ( )

The GetlhFit method

5.20.3.11 string RootFileFormat::GetlhFitInfo (lhFit * *lhFitData*)

The GetlhFitInfo method allow us get the information in free format

5.20.3.12 int RootFileFormat::GetMaxEntries (void)

This method get the maximum entries on the opened file

5.20.3.13 planeFit * RootFileFormat::GetplaneFit ( )

The GetplaneFit method

5.20.3.14 planeFitEven * RootFileFormat::GetplaneFitEven ( )

The GetplaneFitEven method

5.20.3.15 string RootFileFormat::GetplaneFitEvenInfo (planeFitEven * *planeFitEvenData,* int *nHit*)

The GetplaneFitEvenInfo method allow us get the information in free format

5.20.3.16 string RootFileFormat::GetplaneFitInfo (planeFit * *planeFitData,* int *nHit*)

The GetplaneFitInfo method allow us get the information in free format

5.20.3.17 planeFitOdd * RootFileFormat::GetplaneFitOdd ( )

The GetplaneFitOdd method

5.20.3.18 string RootFileFormat::GetplaneFitOddInfo (planeFitOdd * *planeFitOddData,* int *nHit*)

The GetplaneFitOddInfo method allow us get the information in free format

5.20.3.19 stdCuts * RootFileFormat::GetstdCuts ( )

The GetstdCuts method

5.20.3.20 stdCutsEven * RootFileFormat::GetstdCutsEven ( )

The GetstdCutsEven method

5.20.3.21 string RootFileFormat::GetstdCutsInfo (stdCuts * *stdCutsData*)

The GetstdCutsInfo method allow us get the information in free format

Parameters

stdCutsData	retrieves the stdCuts class information

5.20.3.22 stdCutsOdd * RootFileFormat::GetstdCutsOdd ( )

The GetstdCutsOdd method

5.20.3.23 TChain * RootFileFormat::GetTChain ( )

The xcdf-root

5.20.3.24 RootTrig * RootFileFormat::GetXCDFBranchs ( )

The GetXCDFBranch Method

5.20.3.25 void RootFileFormat::Initialize (std::string * , int , int)

This method allows us to initialize the event reader, in a future this will be inside the constructor, in ordet to improve the memory management

Parameters

File	Indicates the file to be read
EventNumber	Inidicate the number of the event we want to start the browsing process

5.20.3.26 void RootFileFormat::NextEvent (void)

This method jump to the immediate next event

5.20.3.27 void RootFileFormat::PreviousEvent (void)

This method jump to the immediate previous event

5.20.3.28 int RootFileFormat::SetArraySize (void)

This method allow us to get the size for the arrays. i. e. {x,y,z}PMT ,etc

5.20.3.29 TEventList * RootFileFormat::SetCuts (std::string *CutString*)

Parameters

CutString	The string that contain the cuts logic

Returns

EventList List of Events (TEventList) that passes the set of cuts logic.

Un Corte de Selección es un proceso que permite obtener una colección de datos, mediante la aplicacion de un conjunto de condiciones que permitan discriminarlos

5.20.3.30 void RootFileFormat::SetRecoBranch (void)

Set the branches and memory for reco files reconstructed by vamos-reader-reco or offline-reco Assign the values to the event class

Assign for the coreGuess class

Assign for the coreGauss class

Assign for the planeFit class

Assign for the planeFitOdd class

Assign for the planeFitEven class

Assign for the lhFit class

Assign for the compactness40 class

Assign for the stdCuts class

Assign for the stdCutsOdd class

Assign for the stdCutsEven class

5.20.3.31 void RootFileFormat::SetTrigBranch (void)

Set the Branch for trig root files generated with xcdf-root

5.20.3.32 void RootFileFormat::ToEvent (int *EventNumber*)

This method allow to "jump" to desired EventNumber

Parameters

EventNumber	The disired event number to show.

The documentation for this class was generated from the following files:

- include/sedeb/RootFileFormat.h
- src/RootFileFormat.cpp

5.21 RootTrig Class Reference

```
#include <trig.h>
```

Public Attributes

- unsigned long long **trigversion**
- unsigned long long **trigeventID**
- unsigned long long **trigrunID**
- unsigned long long **trigtimeSliceID**
- unsigned long long **trigtrigger_flags**
- unsigned long long **trigevent_flags**
- unsigned long long **triggtc_flags**
- unsigned long long **triggpsSec**
- unsigned long long **triggpsNanosec**
- unsigned long long **trignHit4Edge**
- unsigned long long **trignHit2Edge**
- long long * **trighitTime_4Edge**
- long long * **trigloTOT_4Edge**
- long long * **trighiTOT_4Edge**
- long long * **trigtime01_4Edge**
- long long * **trigchannelID_4Edge**
- long long * **trigflags_4Edge**
- long long * **trigtriggerFlags_4Edge**
- long long * **trighitTime_2Edge**
- long long * **trigloTOT_2Edge**
- long long * **trigchannelID_2Edge**
- long long * **trigflags_2Edge**
- long long * **trigtriggerFlags_2Edge**
- long long **triglaserTStart**
- long long **triglaserTStop**
- long long **triglaserLightToTanksStart**
- long long **triglaserLightToTanksStop**

5.21.1 Detailed Description

This class is for the trig files converted from xcdf to root

The documentation for this class was generated from the following files:

- include/sedeb/trig.h
- src/RootFileFormat.cpp

5.22 SFNames Struct Reference

Public Attributes

- std::string * **FNames**
- int **ArraySize**
- QStringList **FileList**

The documentation for this struct was generated from the following file:

- include/sedeb/general.h

5.23 Statistics Class Reference

Statistics Class, implementation of the needed Basic Statistics calculation.

`#include <Statistics.h>`

Public Member Functions

- double GetMean (int, double *)
- double GetStd (int, double *)
- double GetVar (int, double *)
- double GetMax (int, double *)
- double GetMin (int, double *)
- double * Normalize (int, double *, double, double)
- int * LInterp (double, double, int *, int *)
- Statistics ()
- ~Statistics ()

5.23.1 Detailed Description

Statistics Class, implementation of the needed Basic Statistics calculation.

Returns

Returns statistics from an array with size number of elements

Author

Cederik De Leon

5.23.2 Constructor & Destructor Documentation

5.23.2.1 Statistics::Statistics ( )

Makes a simple linear interpolation

Statistics class default constructor

5.23.2.2 Statistics::~Statistics ( )

Statistics class default destructor

5.23.3 Member Function Documentation

5.23.3.1 double Statistics::GetMax (int *size*, double * *value*)

Returns the Variance of an array

5.23.3.2 double Statistics::GetMean (int *size*, double * *value*)

In order to get the Mean from an Array

5.23.3.3 double Statistics::GetMin (int *size,* double * *value*)

Returns the Maximun value of an array

5.23.3.4 double Statistics::GetStd (int *size,* double * *value*)

Returns the mean of an array of size n

In order to get the Standar Deviation from an Array

5.23.3.5 double Statistics::GetVar (int *size,* double * *value*)

Returns the Standard Deviation of an array

5.23.3.6 int * Statistics::LInterp (double *Xp,* double *Length,* int * *InitColor,* int * *EndColor*)

Return a normalized vector, this is the test implementation

This method allow se to retunr the color corresponding in a certain point beetwen color init point and color end point

Parameters

Xp	desired related color point
InitColor	a 3element vector R, G, B related to the init color in gradient
EndColor	a 3element vector (R, G, B) related to the Endo color in gradien
Length	The length of the box containing the color Gradient

5.23.3.7 double * Statistics::Normalize (int *size,* double * *value,* double *C,* double *k*)

Returns the minimun value of an array

This method allow us to recalculate a normalized value to 1 + C, useful for graphics and canvas process in order to avoid non-positive values or zeroes.

Parameters

C	specifies the total amount of shifting, use it when presume non-positives vector values
k	specifies the proportionality constant, use it to scale or rescale vector

Value Shift, recorrer

The documentation for this class was generated from the following files:

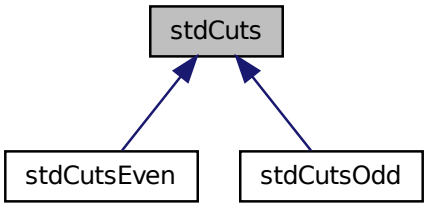
- include/sedeb/Statistics.h
- src/Statistics.cpp

5.24 stdCuts Class Reference

a class to store the information stored on the ROOT file format

`#include <stdCuts.h>`

Inheritance diagram for stdCuts:



Public Member Functions

- stdCuts ()

Public Attributes

- unsigned long long **nHit**
- unsigned long long nCh
- unsigned long long nTank
- unsigned long long * isSelected

5.24.1 Detailed Description

a class to store the information stored on the ROOT file format

The stdCuts Class for the EBViewer

Author

Cederik De León

5.24.2 Constructor & Destructor Documentation

5.24.2.1 stdCuts::stdCuts ( )

The stdCuts constructor

5.24.3 Member Data Documentation

5.24.3.1 unsigned long long* stdCuts::isSelected

the total number of tanks on the stdCuts from an Event

5.24.3.2 unsigned long long stdCuts::nCh

the total number of hits inside the stdCuts from an Event

5.24.3.3 unsigned long long stdCuts::nTank

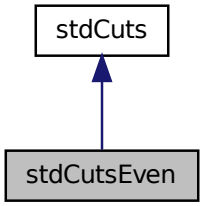
the total number of channels active on the stdCuts from an Event

The documentation for this class was generated from the following files:

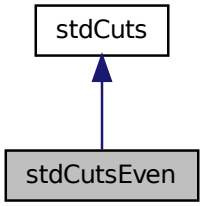
- include/sedeb/stdCuts.h
- src/RootFileFormat.cpp

5.25 stdCutsEven Class Reference

Inheritance diagram for stdCutsEven:



Collaboration diagram for stdCutsEven:



Public Member Functions

- stdCutsEven ()

Additional Inherited Members

5.25.1 Constructor & Destructor Documentation

5.25.1.1 stdCutsEven::stdCutsEven ( )

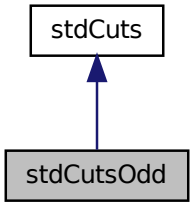
The stdCutsEven constructor

The documentation for this class was generated from the following files:

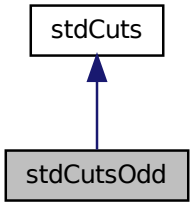
- include/sedeb/stdCuts.h
- src/RootFileFormat.cpp

5.26 stdCutsOdd Class Reference

Inheritance diagram for stdCutsOdd:



Collaboration diagram for stdCutsOdd:



Public Member Functions

- stdCutsOdd ()

Additional Inherited Members

5.26.1 Constructor & Destructor Documentation

5.26.1.1 stdCutsOdd::stdCutsOdd ( )

The stdCutsOdd constructor

The documentation for this class was generated from the following files:

- include/sedeb/stdCuts.h
- src/RootFileFormat.cpp

Index