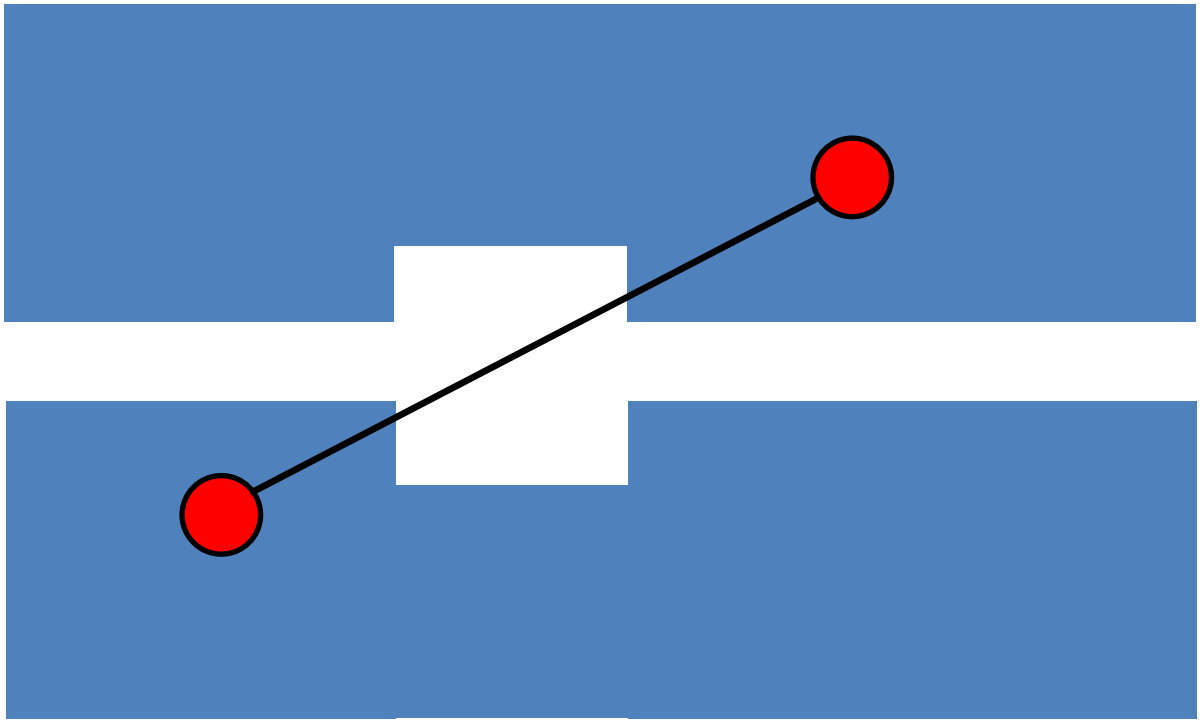
Positron Trigger

HPS Spring Collaboration Meeting 2018

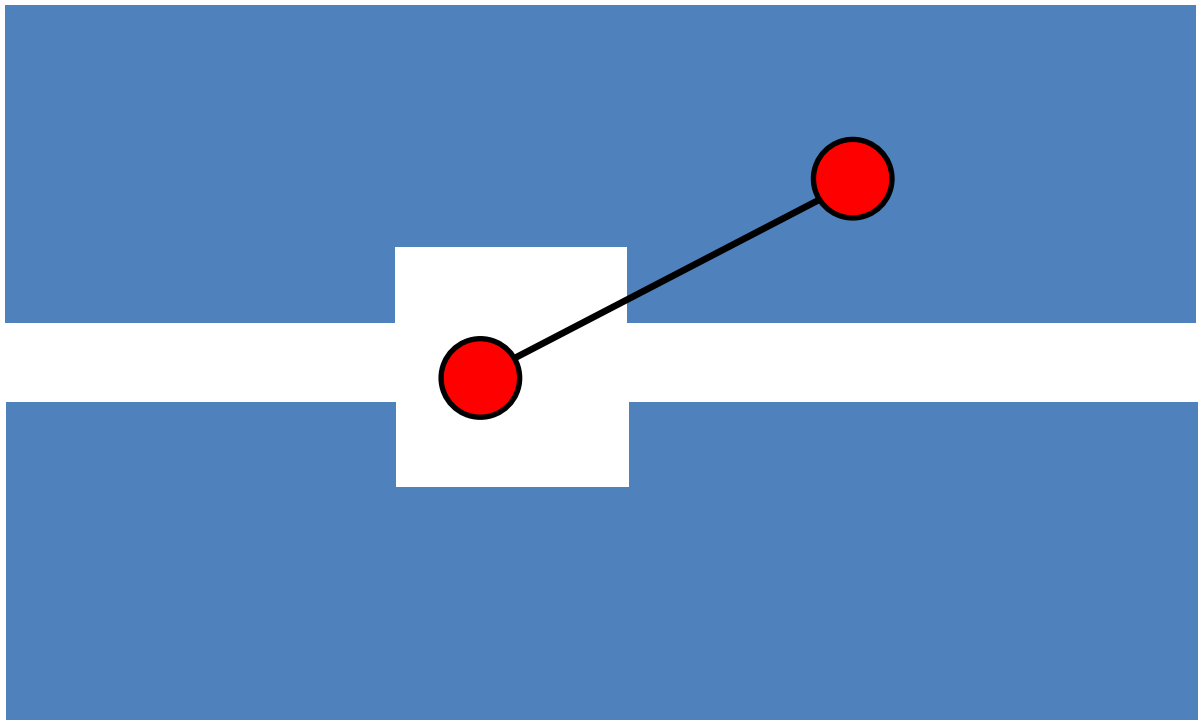
Introduction

- The HPS experiment has thus far employed a pair trigger for the purpose of selecting trident-like events.
  - The pair trigger takes two calorimeter clusters and, using a number of cuts, selects those most likely to be tridents.



Introduction

- It turns out that for many tridents, the electron in the pair is lost to the beam hole and, further, wide-angle bremsstrahlung rates are very high.
  - Many lost electrons are good, high-energy particles.
  - Wide-angle bremsstrahlung can often cause a pair trigger.



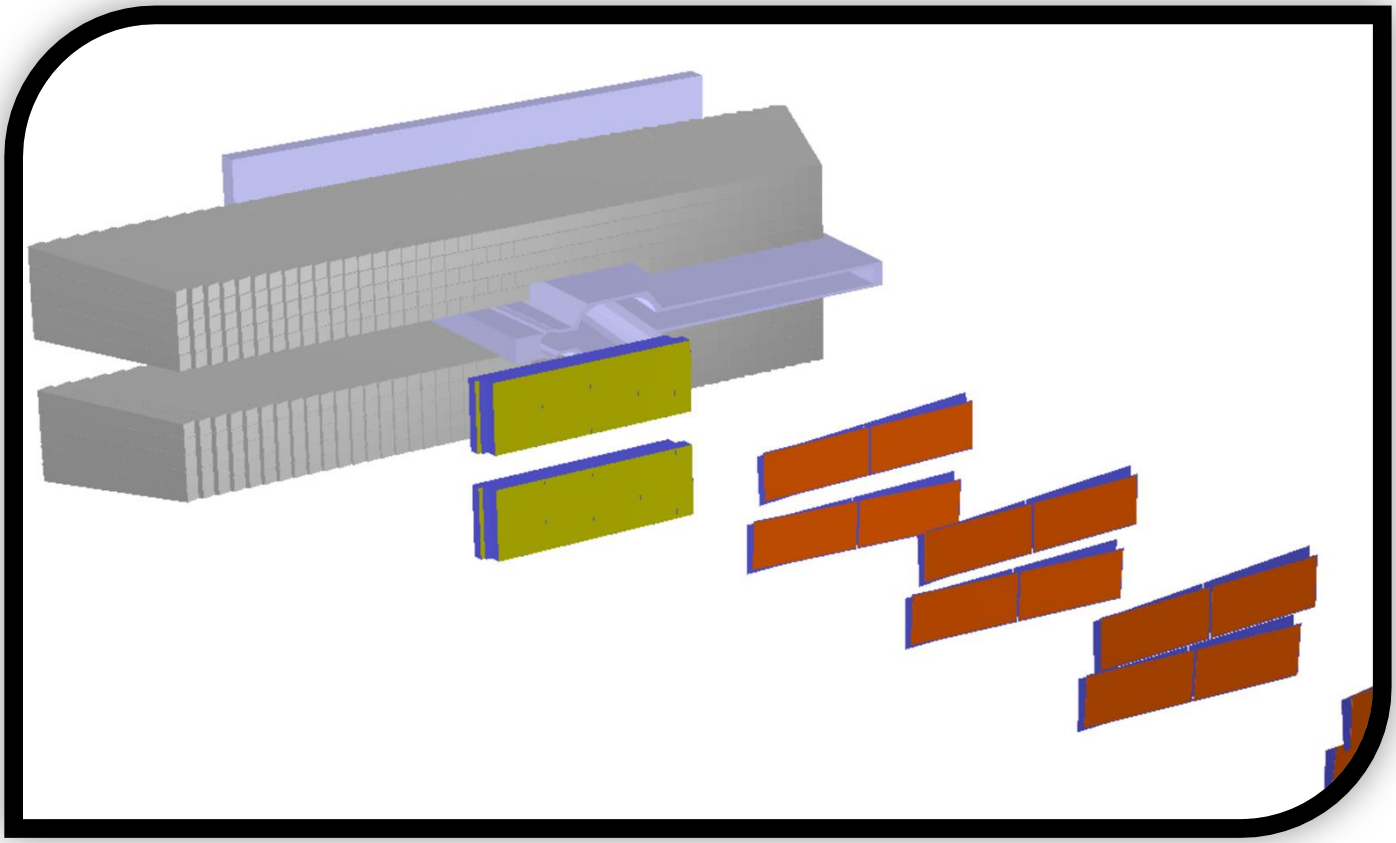
Introduction

- A possible solution to this is positrons. Positrons are generally only seen as part of a trident.
  - A positron trigger should be able to select events where the electron is lost, increasing trident rates in readout.
  - A positron trigger should be more pure and have fewer WAB events.

- A singles triggers won't work!
  - Rates are extremely high, even when limited to just the positron side of the detector.
  - There is not a reliable way to differentiate a positron from other particles using only the calorimeter.
  - The SVT can not perform reconstruction quickly enough to be used for triggering purposes.

Introduction

- To catch these positrons, we create a hodoscope that sits between the SVT and the calorimeter on the positron side only.



Hodoscope Software Implementation

- To better study the effects of a hodoscope and positron trigger, it is important to implement the hodoscope for usage in HPS-Java.
- This is complete!
- The geometry can be automatically generated using code in HPS-Java.
- This code is responsible for generating specifically the hodoscope geometry, and handles the following components:
  - **Scintillators**: The actual active components of the detector.
  - **Reflectors**: Reflective material that surrounds the sides of the scintillators. (Blue in the above picture.)
  - **Covers**: Material that covers the front and back of the hodoscope crystals. (Gold in the above picture.)
  - **Buffer**: Material that sits between the first and second hodoscope layers.

Hodoscope Software Implementation

- By using this method, the hodoscope geometry may be easily inserted into new or existing detectors using only an XML declaration in the detector's compact.xml.

```
<detector id="N" name="Hodoscope" type="Hodoscope_v1" readout="HodoscopeHits"
            insideTrackingVolume="true" vis="HodoscopeVis">
      <scintillator_material value="Polystyrene" />
      <cover_material value="TitaniumDioxide" />
      <reflector_material value="Mylar" />
      <buffer_material value="Carbon" />
</detector>
```

- The hodoscope XML element is designed to offer a great deal of flexibility, and minimize the need for users to interface with the underlying code.

Hodoscope Software Implementation

- Only a few arguments are mandatory when declaring the hodoscope XML for a detector. These are the component material definitions:
  - **Scintillator Material:** Defines the material for the actual hodoscope crystals.
  - **Cover Material:** Defines the material for the cover that goes in front of and behind the hodoscope crystals.
  - **Reflector Material:** Defines the material for the side reflectors.
  - **Buffer Material:** Defines the material for the layer buffer.
- If not other options are declared, the hodoscope will use preset “nominal” values.
  - These will be set more formally once the hodoscope is physically implemented and a more final nominal set-up is defined.

Hodoscope Software Implementation

- However, virtually every aspect of the hodoscope design may be declared through the XML declaration. These include:
  - **Positioning:** It is possible to declare the x- and y-position of the first (inner-most) crystal of each layer of the hodoscope, both top and bottom.
    - Other crystal positions are determined by the dimensions of the various components and placed relative to the first crystals.
    - The z-position of each layer may also be declared, but is always the same for the top and bottom.
    - The x-position of the inner edge of the layer buffer may also be declared.

Hodoscope Software Implementation

- However, virtually every aspect of the hodoscope design may be declared through the XML declaration. These include:
  - **Component Dimensions:** Most component dimensions may be declared in the XML description.
    - The dimensions of the scintillators may be declared freely. The depth and height of all crystals is always the same, but the widths can be specified for each crystal.
    - The thickness of both the crystal reflectors and the crystal covers may be defined.
    - The thickness, width, and depth of the layer buffer may be defined.
  - The geometry code will automatically position the component elements in the appropriate positions relative to the defined starting crystal positions to account for the declared dimensions.

Hodoscope Software Implementation

- However, virtually every aspect of the hodoscope design may be declared through the XML declaration. These include:
  - **Crystal Set-Up:** Lastly, the number of crystals can be modified through the XML.
    - Crystal widths are given in a comma-delimited list in the XML.
    - Each entry is treated as a new crystal.
    - Crystals are automatically generated with covers and reflectors, and placed one after another in the order of their declaration after the starting crystal.

- This set-up allows the user to easily modify almost any part of the hodoscope design as desired without the need to find and modify the underlying generating code.

Hodoscope Software Implementation

- When implemented, SLIC will output hit objects into the data representing energy depositions onto the hodoscope.
- These are created in the form of `SimCalorimeterHit` objects in the collection “HodoscopeHits”.
- Hodoscope simulation hits contain a number of details about the hit. Most notably:
  - Energy: The energy deposition that created the hit.
  - Time: The time, with respect to the beam bunch, at which the energy was deposited.
  - Truth Information: Which MC particle was responsible for creating the hit, and how much energy they deposited.
  - Additionally, position is recorded. The hit positions will always report the same y- and z-values for a given crystal, but are further segmented along the x-direction to give a more precise x-position.

Hodoscope Software Implementation

- Presently, no detector response is emulated. Rather, hodoscope MC is analyzed using the pure truth information.
  - It is expected that a more detailed simulation can be created later, once the hodoscope is physically built and its in-practice behavior can be better understood.

- Instead, all truth hits within a beam bunch (2 ns window) are added together and treated as a single object, with the combined energy of the truth hits.

First-Pass MC Analysis

- With a detector geometry defined and working, Monte Carlo analysis of the hodoscope and positron trigger has begun.

- When designing a positron trigger, we want to maximize the selection of analyzable events – i.e., events which contain enough of the right kind of data from which a usable trident may be reconstructed.

- A major component of any analyzable event is that it contains a positron and electron track, but this information is not available until reconstruction, which requires that we run the readout simulation first.

First-Pass MC Analysis

- To get around this issue, we create a “dumb positron trigger” which performs readout on any event that contains a cluster in the positron side of the calorimeter.
  - All proposed positron trigger plans will require this condition to be met anyway, so this does not exclude any potential triggerable events.
  - By using pure trident Monte Carlo, we also ensure that we only consider desirable events and don’t have to worry about junk triggers.

- This new readout data may then undergo reconstruction, and trigger tuning can account for which events include useful tracks.

First-Pass MC Analysis

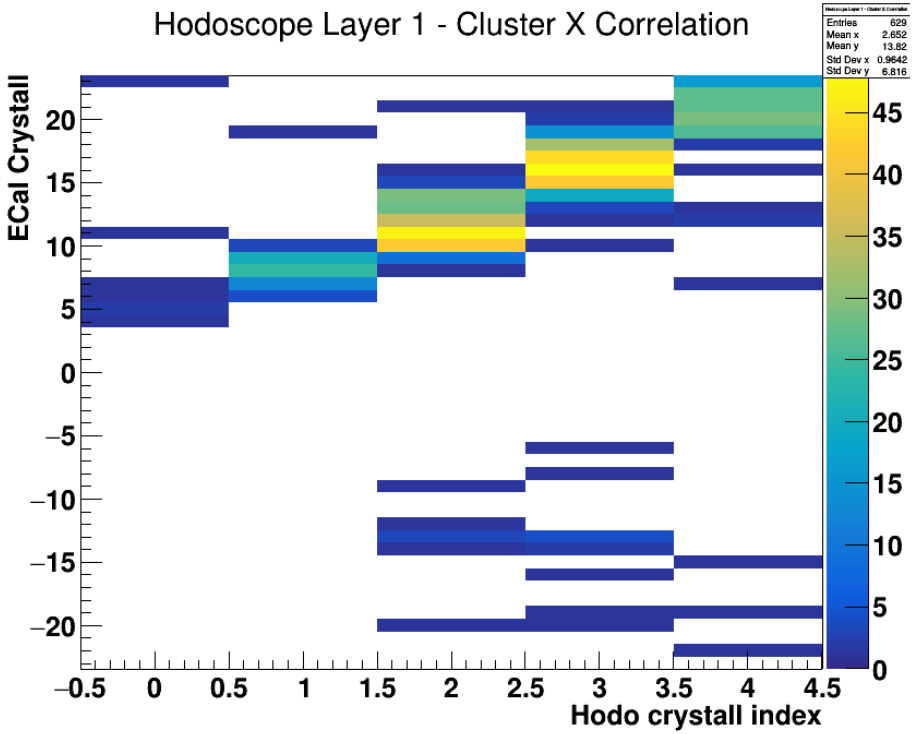
- Once reconstruction is complete, the first-pass analysis may begin. The goal of this step is to establish the necessary relations to define the positron trigger.
- The idea behind the positron trigger is as follows:
  - When a positron occurs, it will pass through the hodoscope and then impact the calorimeter face.
  - The positron should pass through both layers of the hodoscope in a consistent fashion.
  - There should be some consistent relation between the position of the positron in the hodoscope versus the position of the positron on the calorimeter face.
  - By requiring that the detector signature be consistent with all three of these points, non-positronic events can be strongly excluded at the trigger level.

First-Pass MC Analysis

- To establish these relations, we first start by selecting only “analyzable” events. These are events which:
  - Have one positive and one negative track.
  - One track must point upwards, and one track must point downwards (i.e. a top and bottom track).
- We then aim to establish a relation between hodoscope hits in each layer.
  - To do this, we take advantage of the truth information stored in the hodoscope hits.
  - We can determine which MC particle created a given hodoscope hit, and use this generating particle to match hodoscope hits on different layers.
  - We plot the x- and y-indices of hodoscope hits on layer 1 with those of layer 2, and use this plot to generate a correlation table between the two layers.

First-Pass MC Analysis

- The same process can then be performed with the calorimeter clusters.
  - New modifications made to the (currently iss166) readout allows for full propagation of calorimeter truth information.
  - This means that it is also possible to link hodoscope hits with clusters created by the same particle, and a similar plot made for each layer of the hodoscope versus calorimeter cluster position.
  - Correlation tables are again generated.



First-Pass MC Analysis

- The correlation tables are presented below.

Table 1:

Layer 1 Index	Layer 2 Min Index	Layer 2 Max Index
0	0	1
1	0	1
2	1	2
3	2	3
4	3	4

Table 2:

Hodoscope Layer 1 Hit			Hodoscope Layer 2 Hit		
Hodo Index	Cl. Min Index	Cl. Max Index	Hodo Index	Cl. Min Index	Cl. Max Index
0	4	7	0	6	7
1	6	6	1	7	12
2	8	10	2	11	17
3	12	21	3	14	21
4	18	23	4	19	23

Trigger Proposal

- From these correlations, a first-pass trigger is selected.
  - A hodoscope hit must exist in each layer of the hodoscope with $E \geq 1\ \mathrm{MeV}$.
  - A cluster must exist in the calorimeter with $E \geq 200\ \mathrm{MeV}$.
  - There must exist a pair of two hodoscope hits that meet the relation requirements in Table 1, where both hits also meet the relation requirements in Table 2.

- As a first-pass test of these trigger rules, we monitor how many analyzable events are excluded at each step.

Trigger Proposal

- The results of this test are below:

Event Type	Event Count	Percentage Loss
Analyzable Track Pair	118,862	0.00%
A Hodoscope Hit	117,801	0.89%
A Hodoscope Hit per Layer	116,348	2.11%
A Correlated Hodoscope Hit Pair	113,556	4.46%
A Correlated Hodoscope/Cluster Pair	108,839	8.43%

- It appears that the trigger relations retain over 90% of analyzable events which are possible to detect.
- Further optimization may be possible for these tables, particularly for the final hodoscope/calorimeter correlation.
- Further testing will be needed to check trigger performance on full background simulation and perform rate estimation.

Summary

- A hodoscope detector has been implemented into HPS-Java.
  - The hodoscope geometry is generated programmatically, and inserted into detector definitions through the compact.xml.
  - It allows for a significant amount of customization without alteration of the underlying code.

- Monte Carlo trigger simulation of the positron trigger is on-going.
  - A potentially viable trigger has been selected and shows promise in initial testing.
  - More studies are needed before finalization can occur.