Analysis program

- Define Framework and sign up analysis modules
- Read in analysis settings (cuts, analysis parameters, subscribe for plots)
- Set modules' parameters (input run file(s), cuts, plots to draw, etc.)
- framework->beginJob() - initialize framework
- framework->proceed() - process events
- framework->endJob() - sum up, plot and store results

Analysis Modules (1)

DetectorModule *responsible for reconstruction*

- **beginJob**
  - *Set cut parameters*
  - *Initialize histograms*
- **beginRun**
  - *Load calibrations for run (see next slides)*
- **event**
  - *event rejection using cuts*
  - *reconstruction: digits -> physics info*
  - *fill histograms*
- **endRun**
- **endJob**
  - *Plot histograms*

Analysis Modules (2)

InputModule *responsible for reading raw data and switching runs*

beginJob

- *Map data into memory according to data connection file*
- *Initialize input stream*

beginRun

- *pass time of the run (=time stamp of the first event) to next modules*

event

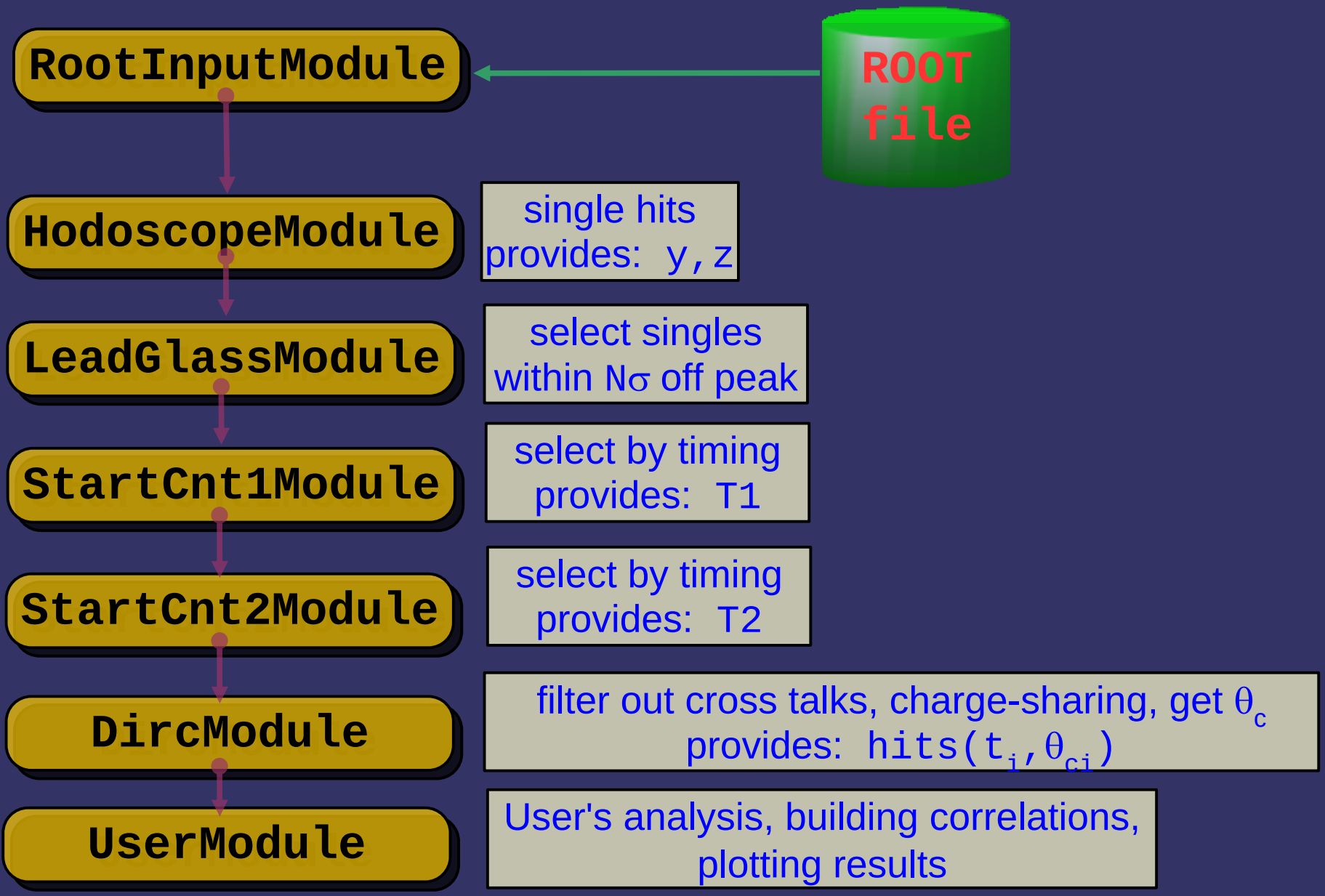
- *read next event from run file*
- *pass raw event data to next modules for reconstruction*
- *notify Framework when next run starts (detected by appearance of a large enough gap between events)*

endRun

endJob

- *Close input stream*

Framework for Endstation A



Calibration lookup table

```
class CalibrationTable {
    CalibrationTable (directory,file_name_pattern);
    int Read (directory,file_name_pattern);
    string GetCalibrationWithTime (time_t t,time_t*,time_t*);
    void Print ();
};
```

Creates lookup table from files in directory matching file_name_pattern. Can manage files of any type.

Given a run/event time stamp one can get a particular calibration file name along with beginning and end of validity period.

CalibrationRange

```
class CalibrationRange {
    CalibrationRange (time_t tbeg,time_t tend);
    bool IsValid (time_t t);
    bool Flawed ();
};
```

Abstract class to enable validating calibration.

Detector Calibrations (1)

```
class HodoscopeCalibration_: public CalibrationRange
{
    short hAth[16], vAth[16]; // ADC thresholds
    // constructor to make up new calibration
    HodoscopeCalibration ();
    // constructor for reading calibration from file
    HodoscopeCalibration (filename,time_t tbeg,time_t tend);
    // store calibration in an XML file
    int Write (filename, time_t tstamp);
};
```

Reading/writing interface is common for every detector.

Detector Calibrations (2)

```
class LeadGlassCalibration {
    float ped, a1, asig;
};

class StartCounterCalibration {
    struct ChannelData {
        float aped; // ADC pedestal
        short athr; // ADC threshold
        float t0, tsig; // mean&sigma of TDC corrected peak
        float tconv; // TDC's count price, ns/count
        map<float,float> adccor; // ADC correction table
    };
    struct ChannelData mcp[4], quantacon;
    // Apply ADC correction to timing: tdc_cor=tdc-adccor+t0
    float GetCorrectedTDC (int ch,float adc,float tdc);
};
```

Detector Calibrations (3)

```
class DircCalibration {
  struct SlotData {
      int ndefined; // Number of defined pads for the slot
      float t0[64]; // Timing offsets for each pad in counts
      float tconv[64]; // TDC's count price, ns/count
  };
  struct SlotData slot2, slot3, slot4, slot5, slot6;
};
```