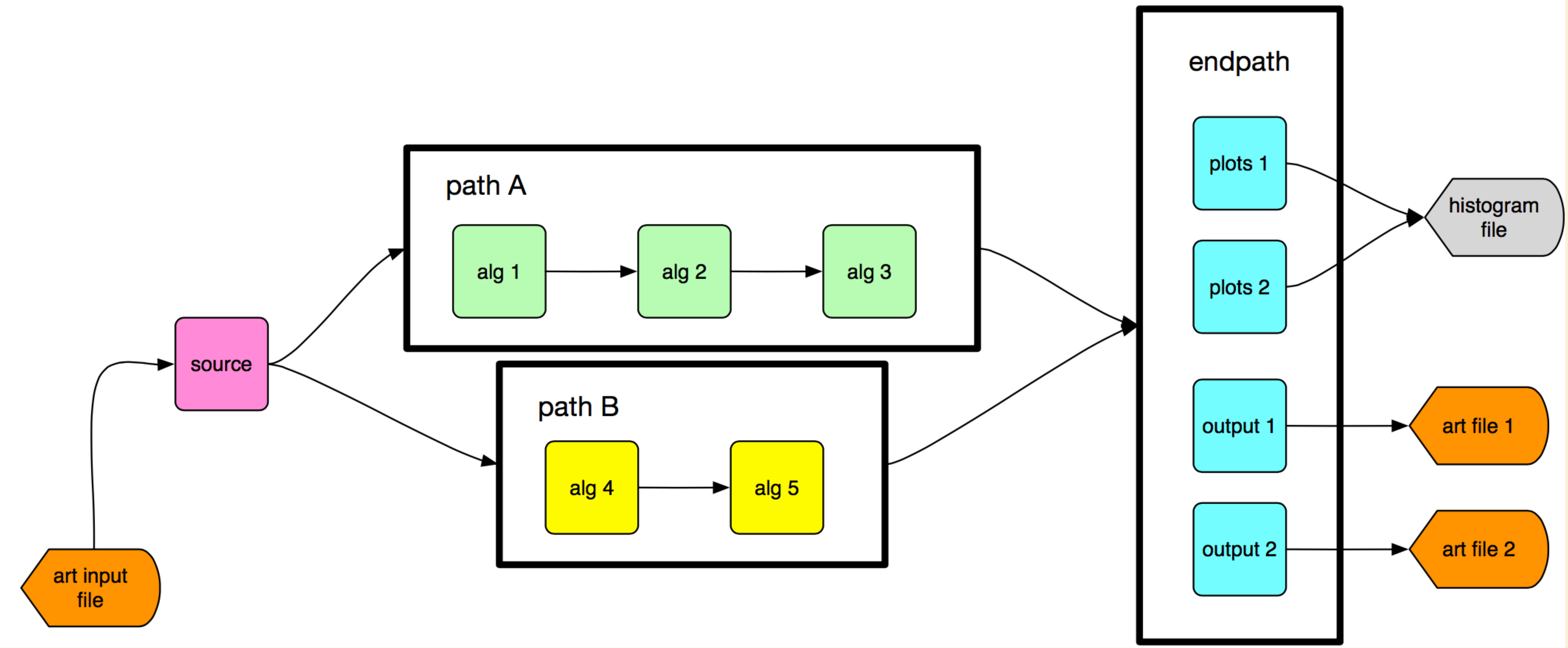
Session 12: *Running multiple modules (Workflow)*

Adam Lyon
art/LArSoft School 2015

Duchamp, *Nude Descending a Staircase, No. 2*, Philadelphia Museum of Art

A Workflow - flow of data and algorithms

The flexibility of art allows you to execute complicated workflows



From Marc's talk on Monday

art Workflow

You can run multiple modules on multiple paths
[Can you think of examples?]

You can slice and dice the workflow:

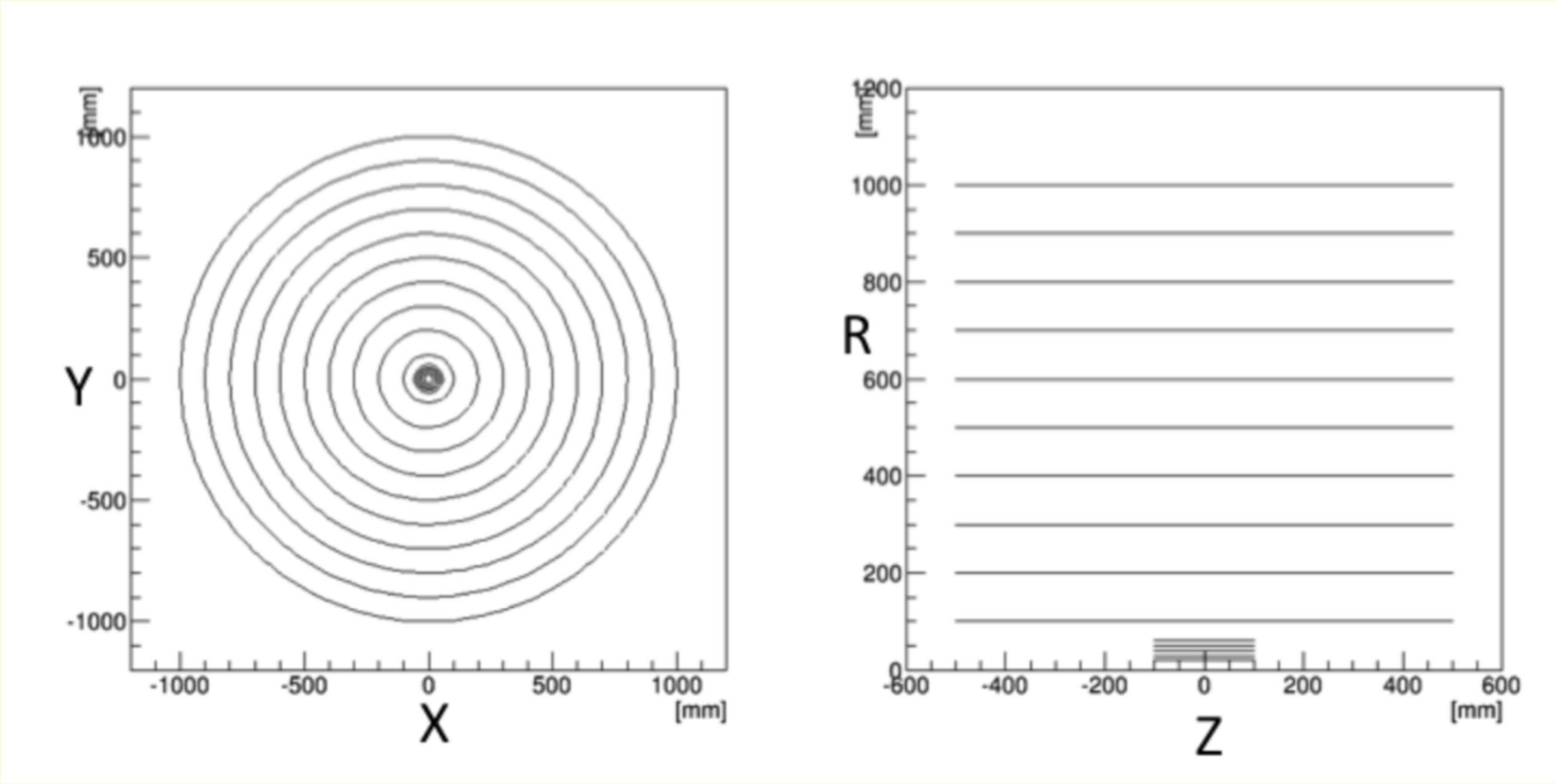
- Do part of the workflow in a job
- Write output to file(s)
- Continue the workflow in a later job with those files as input

[Can you think why you would do this?]

Remember our TOY Detector

You are simulating a "Totally Optimal Yak (TOY) Detector"

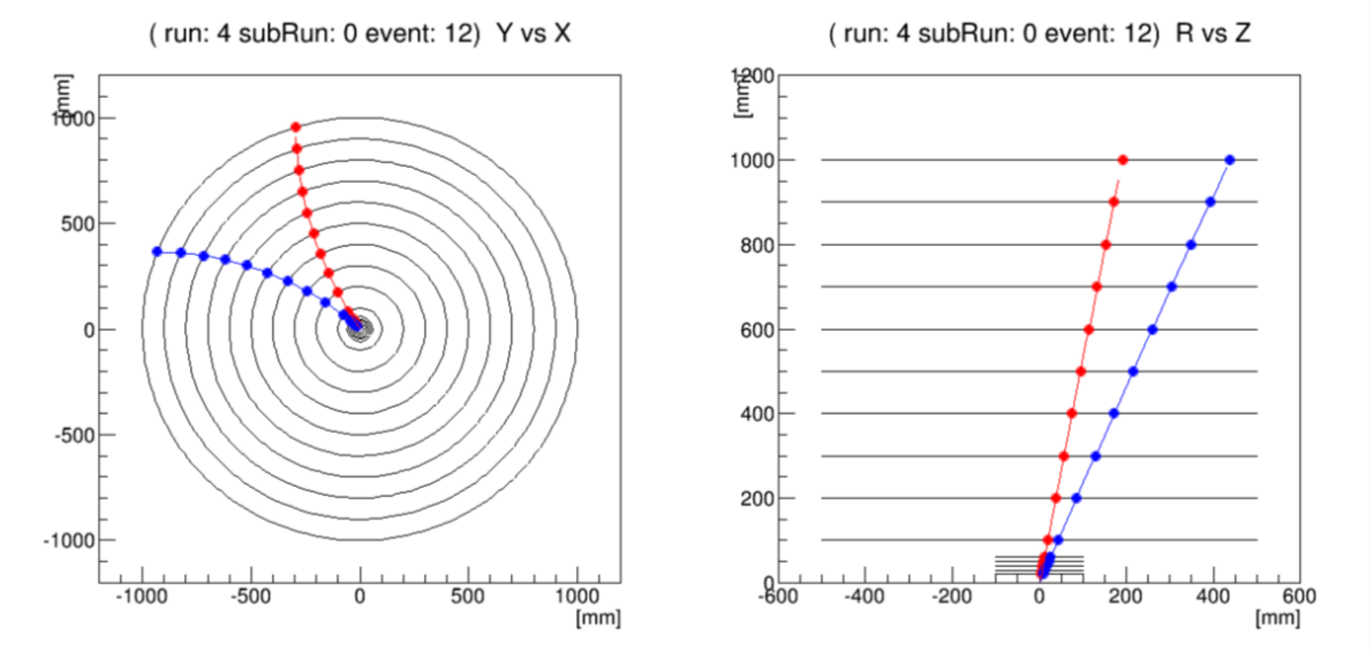
Consider a cylindrical yak



15 concentric massless shells (5 inner closely spaced) in a 1.5 T uniform magnetic field along +z

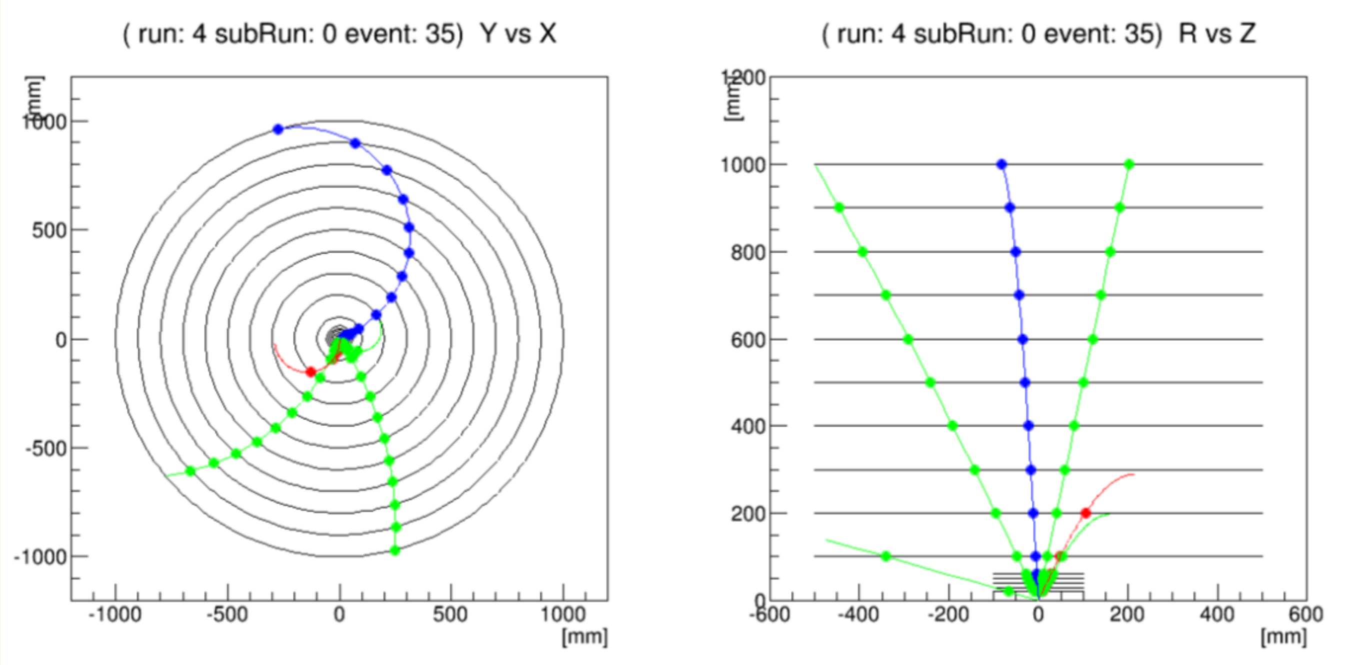
Simple so you can concentrate on software, not physics

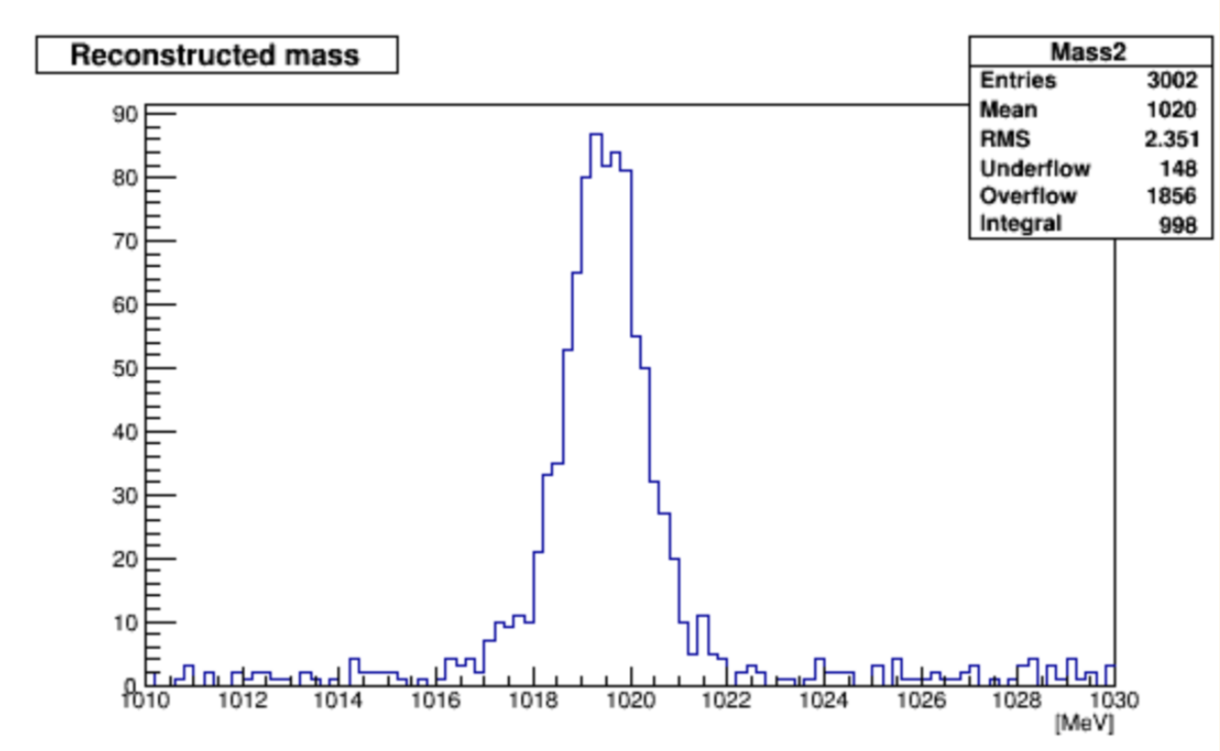
Event generation



$$\phi \rightarrow K^+ K^-$$

with background pion pairs





Let's do a simulation workflow

Go from nothing to fitted tracks from our TOY detector

For each event:

1. Determine the particles in the event (`evtgen`); some events should be signal and some should be background
2. Propagate those particles through the detector and determine true hits (typically with Geant) (`detsim`)
3. Turn the true hits into realistic detector hits (typically called digitization & reconstruction) (`makehits`)
4. Turn the realistic detector hits into fitted tracks (reconstruction & track fitting) (`fitter`)
5. Turn the entire universe into a large set of histograms

Simulation

Never mind how we actually do those steps (that's your physics homework)

The toyExperiment UPS product has code to do all of those things

We're going to see how art is configured to make all that run

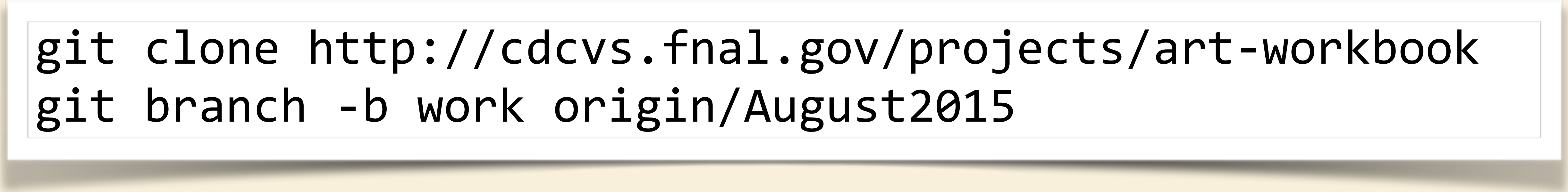
… then we'll see how to change it to do more

Get the code

Log into `alcourse.fnal.gov`

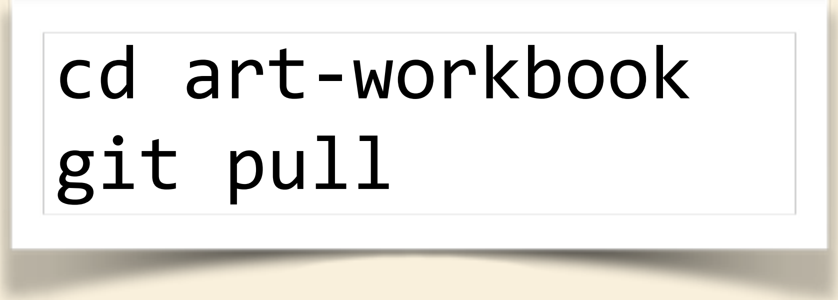
You need the latest version of art-workbook that I changed ~~yesterday~~ a few minutes ago

If you haven't already checked out `art-workbook`



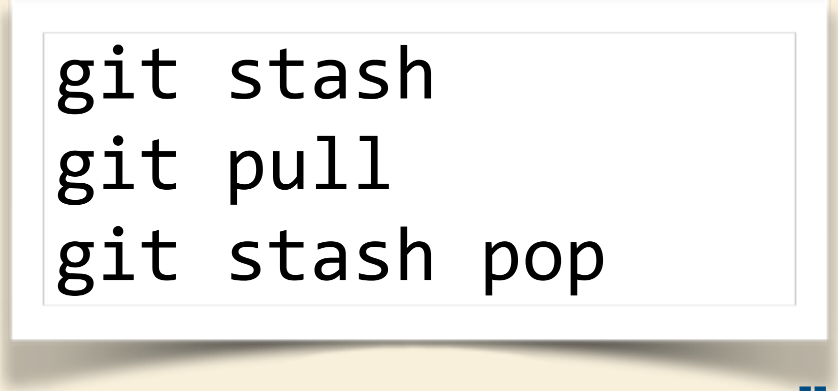
```
git clone http://cdcvs.fnal.gov/projects/art-workbook
git branch -b work origin/August2015
```

If you have checked out `art-workbook`



```
cd art-workbook
git pull
```

if errors:

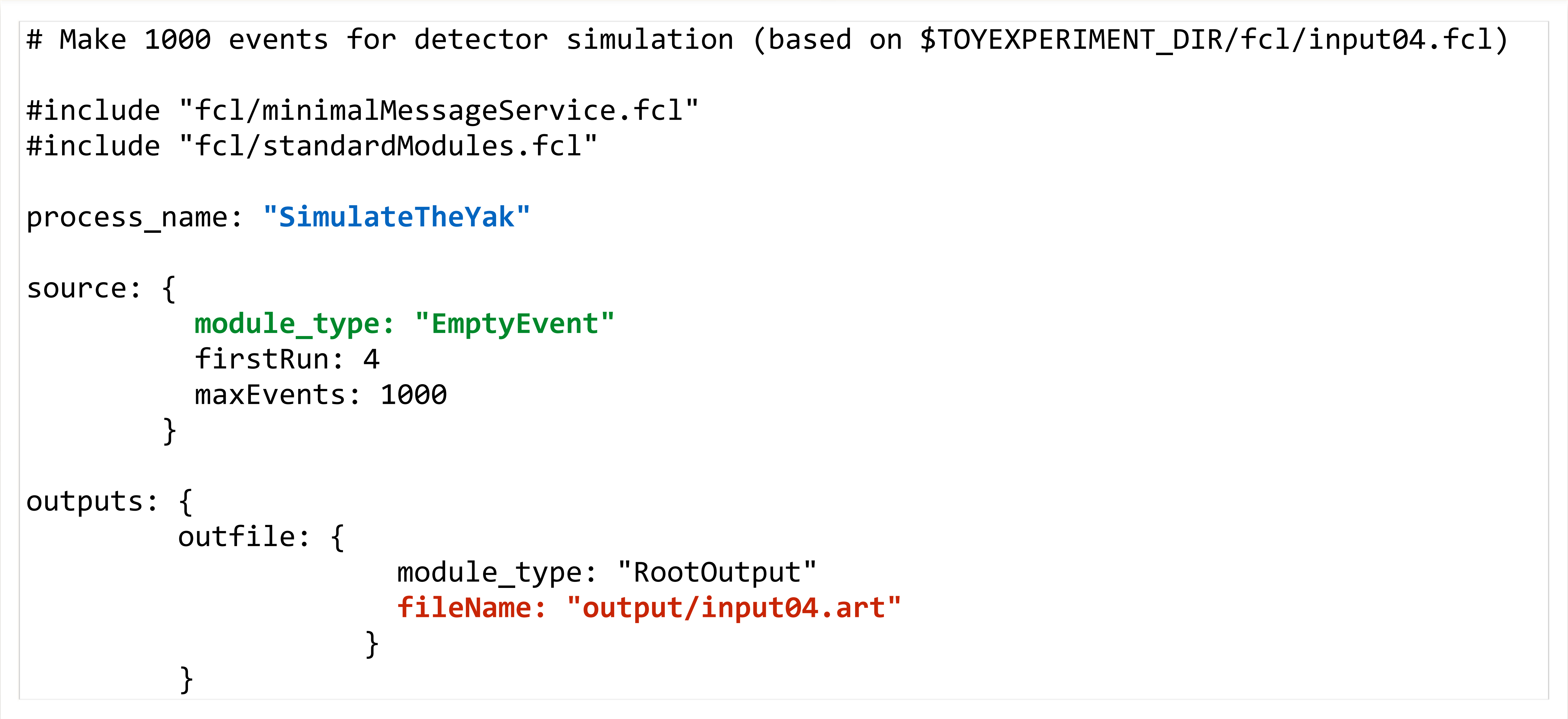


```
git stash
git pull
git stash pop
```

Look at a big workflow

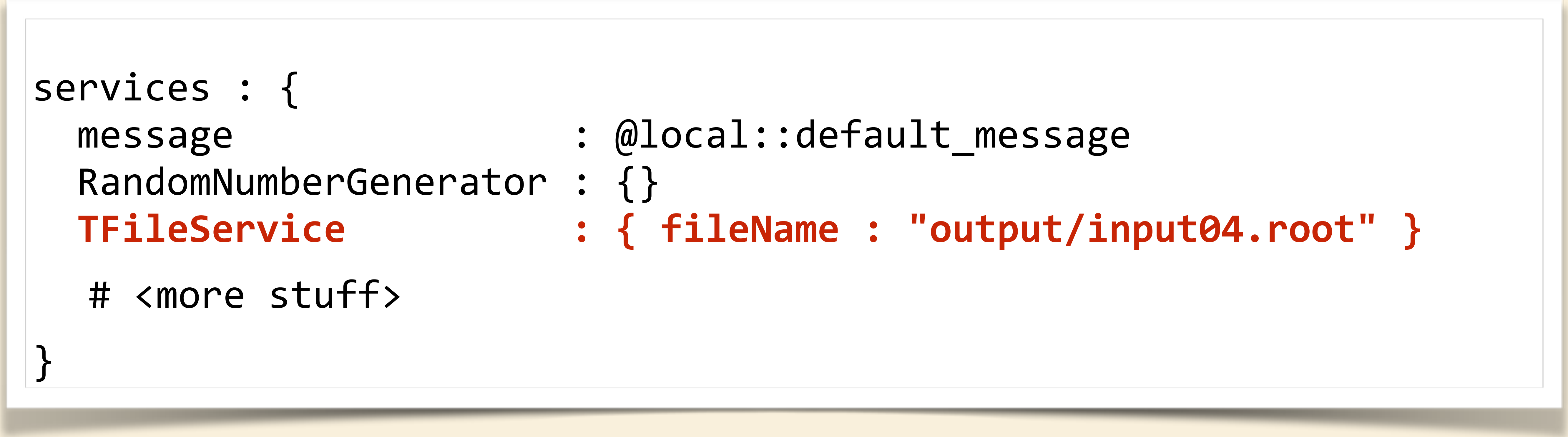
See `art-workbook/MultipleModules/input04_all.fcl`

This is the FCL file that makes 1000 simulated TOY Detector events for the art workbook



```
# Make 1000 events for detector simulation (based on $TOYEXPERIMENT_DIR/fcl/input04.fcl)

#include "fcl/minimalMessageService.fcl"
#include "fcl/standardModules.fcl"

process_name: "SimulateTheYak"

source: {
          module_type: "EmptyEvent"
          firstRun: 4
          maxEvents: 1000
        }

outputs: {
         outfile: {
                      module_type: "RootOutput"
                      fileName: "output/input04.art"
                    }
         }
```

Services part



```
services : {
  message                   : @local::default_message
  RandomNumberGenerator : {}
  TFileService              : { fileName : "output/input04.root" }

   # <more stuff>
}
```

Producers

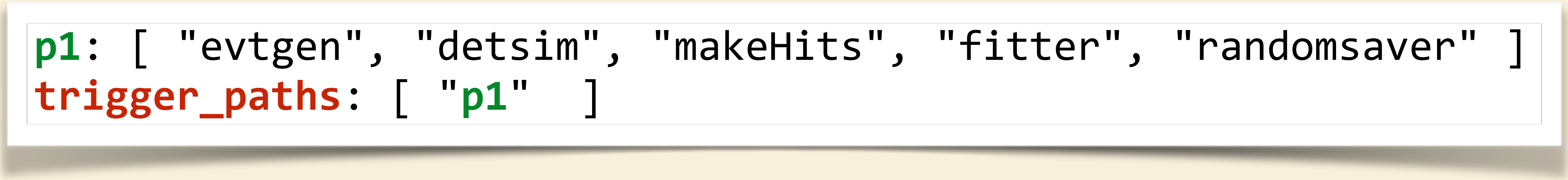
Module label
Module type
Input tag

```
physics: {
 producers: {
  detsim: {
   module_type: "DetectorSimulation"
   genParticleTag: "evtgen"
   seed: 14
  }

  evtgen: {
   module_type: "EventGenerator"
   seed: 13
   signal: {
    pmax: 2000
    pmin: 0
   }
   background: {
    n: 7.5e-1
    pmax: 800
    pmin: 0
   }
  }
```

```
  fitter: {
   module_type: "FindAndFitHelix"
   hitsTag: "makeHits"
   minHits: 5
   seed: 16
  }

  makeHits: {
   module_type: "HitMaker"
   intersectionTags:
    ["detsim:outer","detsim:inner" ]
   seed: 15
  }

  randomsaver: {
   module_type: "RandomNumberSaver"
  }

 } # end of producers
```

Flow of producers

Later in the `physics` section



```
p1: [ "evtgen", "detsim", "makeHits", "fitter", "randomsaver" ]
trigger_paths: [ "p1"  ]
```

`trigger_paths` is a special name

There is one path and the order is as specified

Note the input tags in the producer section

There are also modules called *filters* that are like producers, but return a boolean. On False, the path quits for that event [filters won't be covered here]

Analyzers

```
# Still inside physics {

 analyzers: {

  inspectFits: {
   module_type: "InspectFittedHelix"
   fitsTag: "fitter"
   maxPrint: 0
  }

  inspectGens: {
   module_type:"InspectGenParticles"
   genParticleTag: "evtgen"
   maxPrint: 0
  }

  inspectHits: {
   module_type: "InspectTrkHits"
   hitMakerTag: "makeHits"
   maxPrint: 0

  }
```

```
  inspectIntersections: {
   module_type: "InspectIntersections"
   intersectionTags: ["detsim:outer",
                      "detsim:inner"
   maxPrint: 0
  }

  massPlot: {
   module_type: "MassPlot"
   fitsTag: "fitter"
   maxPrint: 0
  }

 } # end of analyzers
```

Module label

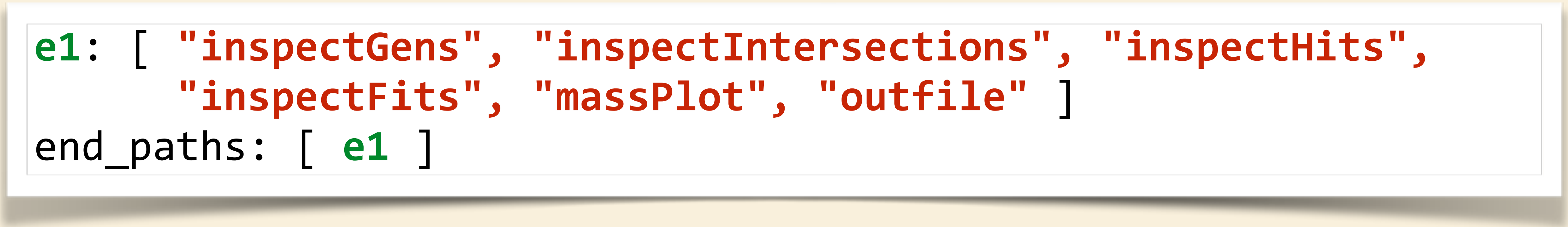
Module type

Input tag

Flow of analyzers/outputs

Analyzers and output modules don’t put data into event, run, sub-run, so their order does not matter

They run for each event after the `trigger_path` completes



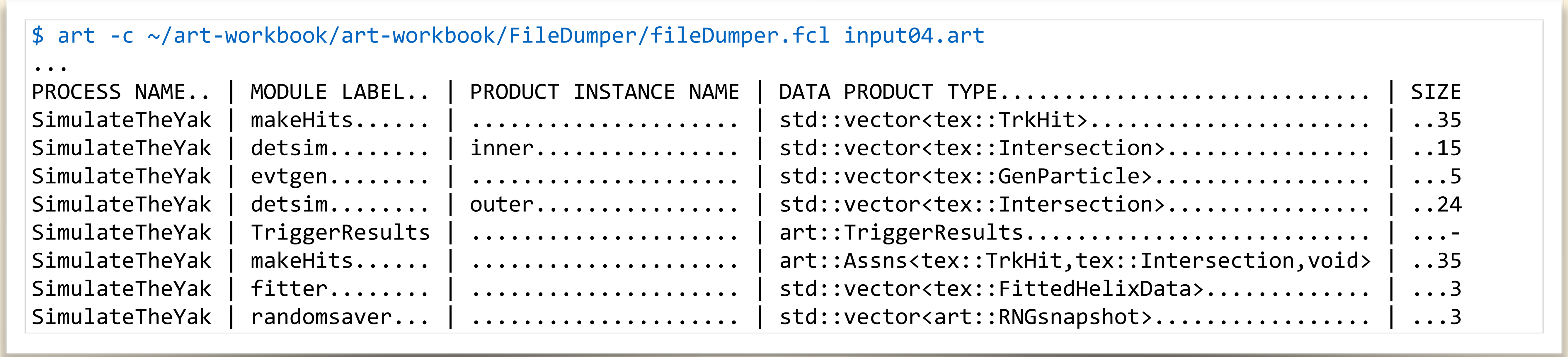
```
e1: [ "inspectGens", "inspectIntersections", "inspectHits",
      "inspectFits", "massPlot", "outfile" ]
end_paths: [ e1 ]
```

Order does not matter

In fact, you don’t have to specify the end_paths, art will figure it out

But specifying `end_paths` will improve readability

What data products get produced?

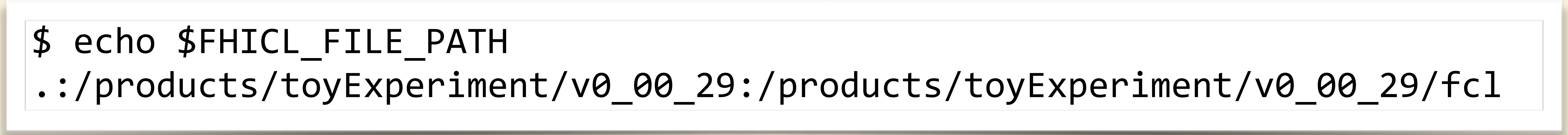


```
$ art -c ~/art-workbook/art-workbook/FileDumper/fileDumper.fcl input04.art
...
PROCESS NAME.. | MODULE LABEL.. | PRODUCT INSTANCE NAME | DATA PRODUCT TYPE............................ | SIZE
SimulateTheYak | makeHits...... | ..................... | std::vector<tex::TrkHit>..................... | ..35
SimulateTheYak | detsim........ | inner................ | std::vector<tex::Intersection>............... | ..15
SimulateTheYak | evtgen........ | ..................... | std::vector<tex::GenParticle>................ | ...5
SimulateTheYak | detsim........ | outer................ | std::vector<tex::Intersection>............... | ..24
SimulateTheYak | TriggerResults | ..................... | art::TriggerResults.......................... | ...-
SimulateTheYak | makeHits...... | ..................... | art::Assns<tex::TrkHit,tex::Intersection,void> | ..35
SimulateTheYak | fitter........ | ..................... | std::vector<tex::FittedHelixData>............ | ...3
SimulateTheYak | randomsaver... | ..................... | std::vector<art::RNGsnapshot>................ | ...3
```

How does art find stuff?

`art -c someFCL.fcl` **and `#include` in FCL files**

Searches `$FHICL_FILE_PATH`



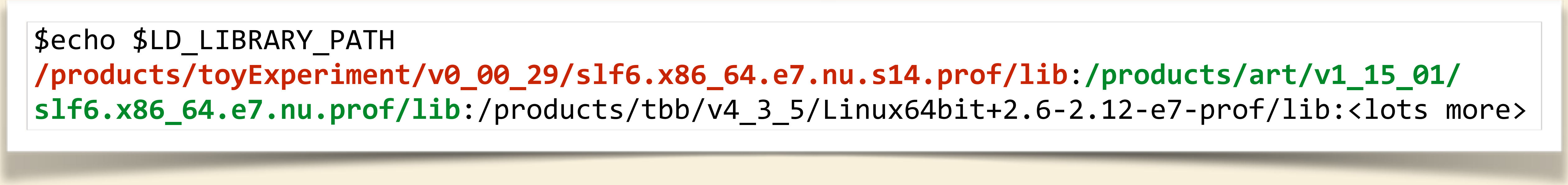
```
$ echo $FHICL_FILE_PATH
.:/products/toyExperiment/v0_00_29:/products/toyExperiment/v0_00_29/fcl
```

Note: products here are UPS products, not data products (I know, it's confusing)

You can use a local FCL with `art -c ./myFCL.fcl` but `#include` statements must still resolve

Searching for libraries

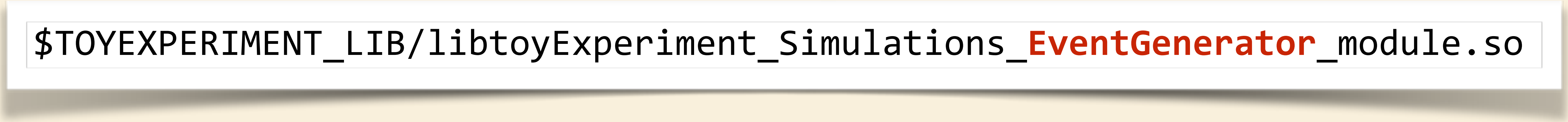
Uses $LD_LIBRARY_PATH [or $DYLD_LIBRARY_PATH]



```
$echo $LD_LIBRARY_PATH
/products/toyExperiment/v0_00_29/slf6.x86_64.e7.nu.s14.prof/lib:/products/art/v1_15_01/
slf6.x86_64.e7.nu.prof/lib:/products/tbb/v4_3_5/Linux64bit+2.6-2.12-e7-prof/lib:<lots more>
```

For example, `module_type: "EventGenerator"`

corresponds to



```
$TOYEXPERIMENT_LIB/libtoyExperiment_Simulations_EventGenerator_module.so
```

$FHICL_FILE_PATH and $LD_LIBRARY_PATH are set up by UPS and build tools (this just happens without you noticing)

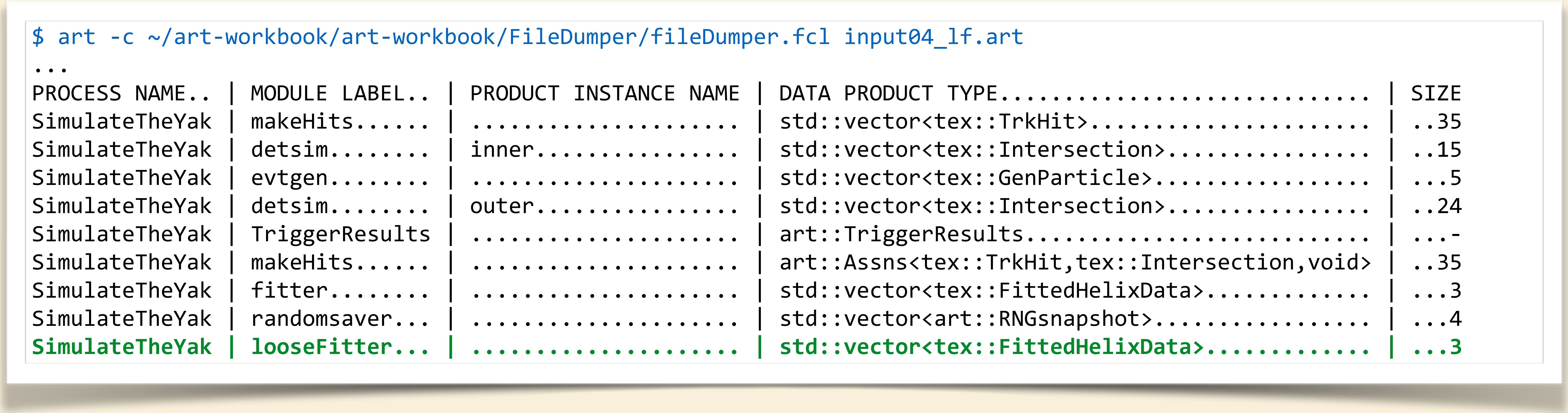
Multiple instances

You learned this in session 9

Add a “Loose Fitter” with 3 minimum hits instead of 5

```
# add to producers
looseFitter : {
   module_type “FindAndFitHelix"
   hitsTag: “makeHits"
   minHits: 3
   seed: 16
}


# add to path
p1: [ "evtgen", "detsim", "makeHits", "fitter", "looseFitter",
      "randomsaver" ]
trigger_paths: [ "p1" ]
```

Check output



```
$ art -c ~/art-workbook/art-workbook/FileDumper/fileDumper.fcl input04_lf.art
...
PROCESS NAME.. | MODULE LABEL.. | PRODUCT INSTANCE NAME | DATA PRODUCT TYPE............................ | SIZE
SimulateTheYak | makeHits...... | ..................... | std::vector<tex::TrkHit>..................... | ..35
SimulateTheYak | detsim........ | inner................ | std::vector<tex::Intersection>............... | ..15
SimulateTheYak | evtgen........ | ..................... | std::vector<tex::GenParticle>................ | ...5
SimulateTheYak | detsim........ | outer................ | std::vector<tex::Intersection>............... | ..24
SimulateTheYak | TriggerResults | ..................... | art::TriggerResults.......................... | ...-
SimulateTheYak | makeHits...... | ..................... | art::Assns<tex::TrkHit,tex::Intersection,void> | ..35
SimulateTheYak | fitter........ | ..................... | std::vector<tex::FittedHelixData>............ | ...3
SimulateTheYak | randomsaver... | ..................... | std::vector<art::RNGsnapshot>................ | ...4
SimulateTheYak | looseFitter... | ..................... | std::vector<tex::FittedHelixData>............ | ...3
```

Multiple paths

In the homework you will split the inner and outer detector reconstruction into two paths

Look at the `makeHits` label and make two of them, one handling inner intersections and one handling outer

You will then need two fitters, one for each set of hits

You will need two paths. For producers before makeHits, just repeat in both paths. Paths must be complete — art knows to only run them once

Adjust the analyzers as well to make separate inner and outer plots

Slicing the workflow

As part of the homework, you will split the workflow into two separate jobs (runs of art)

One with producers and one with analyzers

Make a FCL of only the producers and the output module to produce a file

Make another FCL that reads the file and runs the analyzers to make the histograms

Homework

See 00README.txt in the
`art-workbook/MultipleModules` directory

Don't peek at the answers unless you are really stuck

Good luck!   Ask for help if you need it

Thanks for listening and thanks for learning art!!!