MCViz
Visualizing Monte Carlo for Research and Outreach

http://mcviz.net

Johannes Ebke [1] Peter Waller [2] Tim Brooks [3]

[1]Ludwig-Maximilians-Universität München

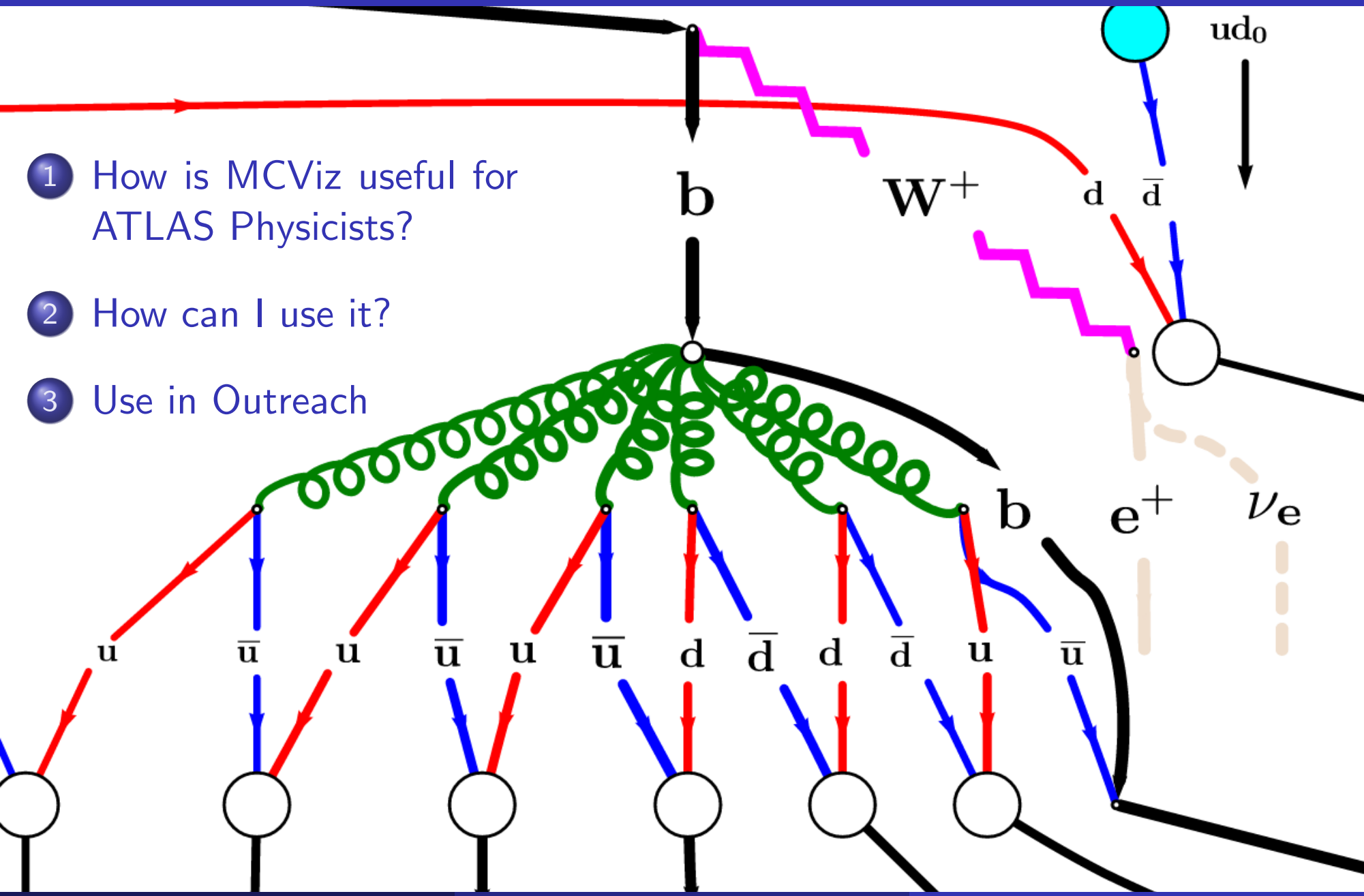
[2]University of Liverpool

[3]Royal Holloway, University of London

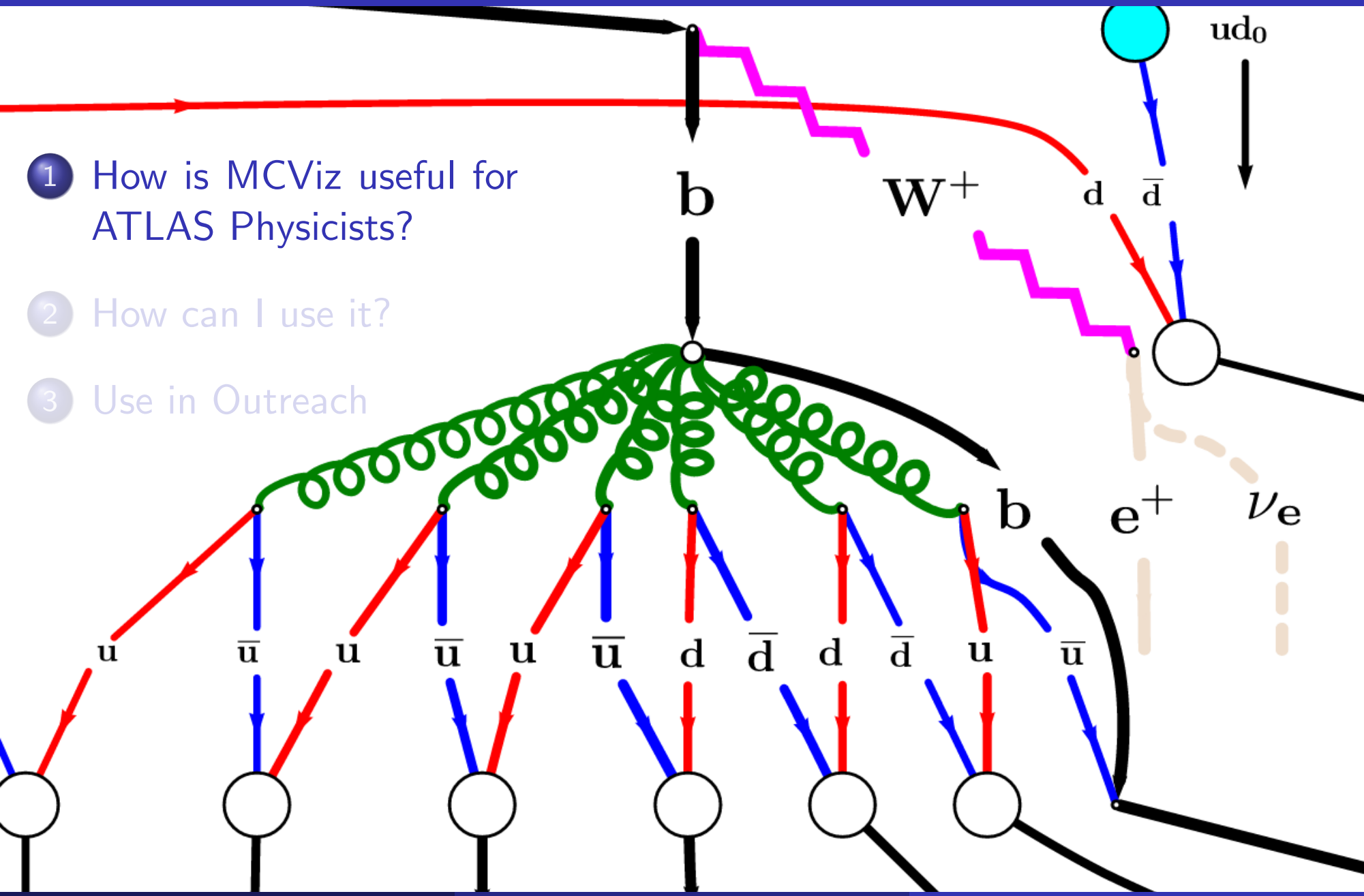
25th of April, 2013

Outline

1. How is MCViz useful for ATLAS Physicists?
2. How can I use it?
3. Use in Outreach



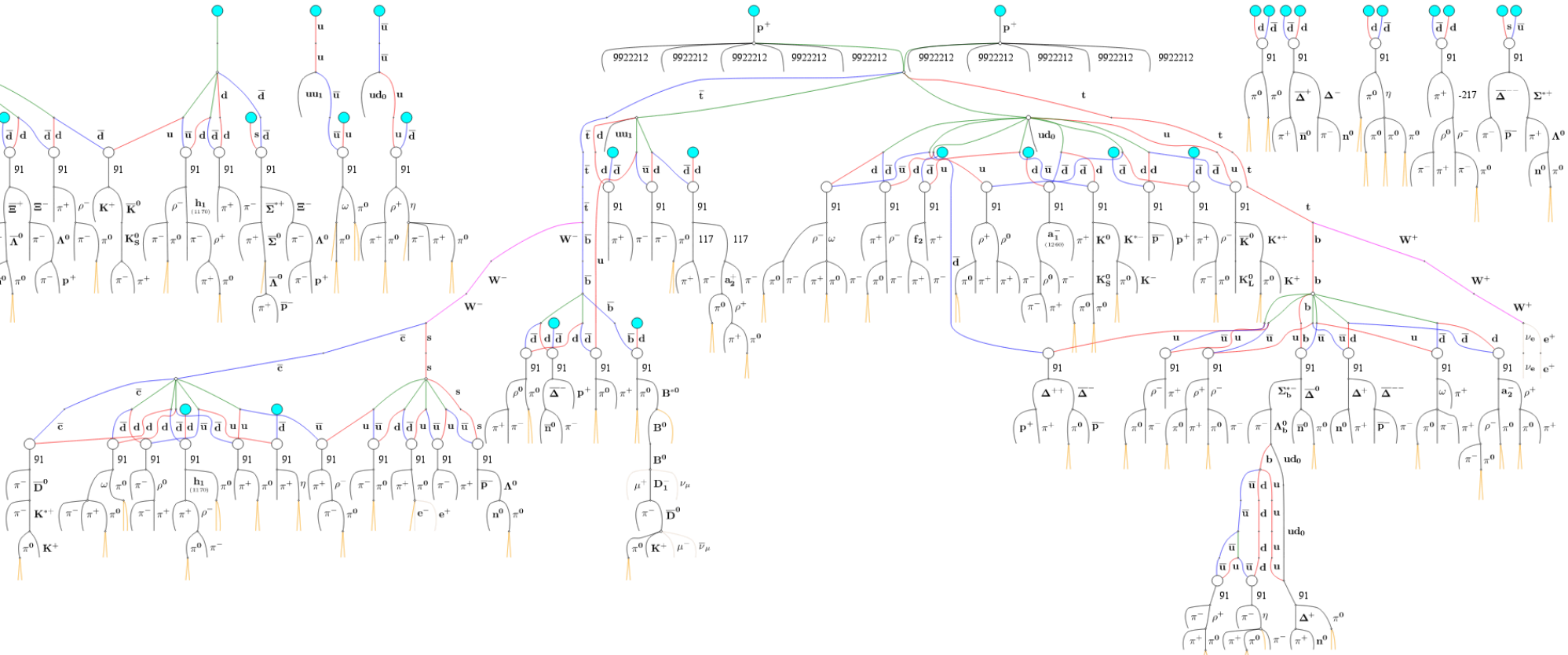
Outline



Quick look at event graphs

Get MCViz from http://mcviz.net and run `./mcviz_bootstrap.py`

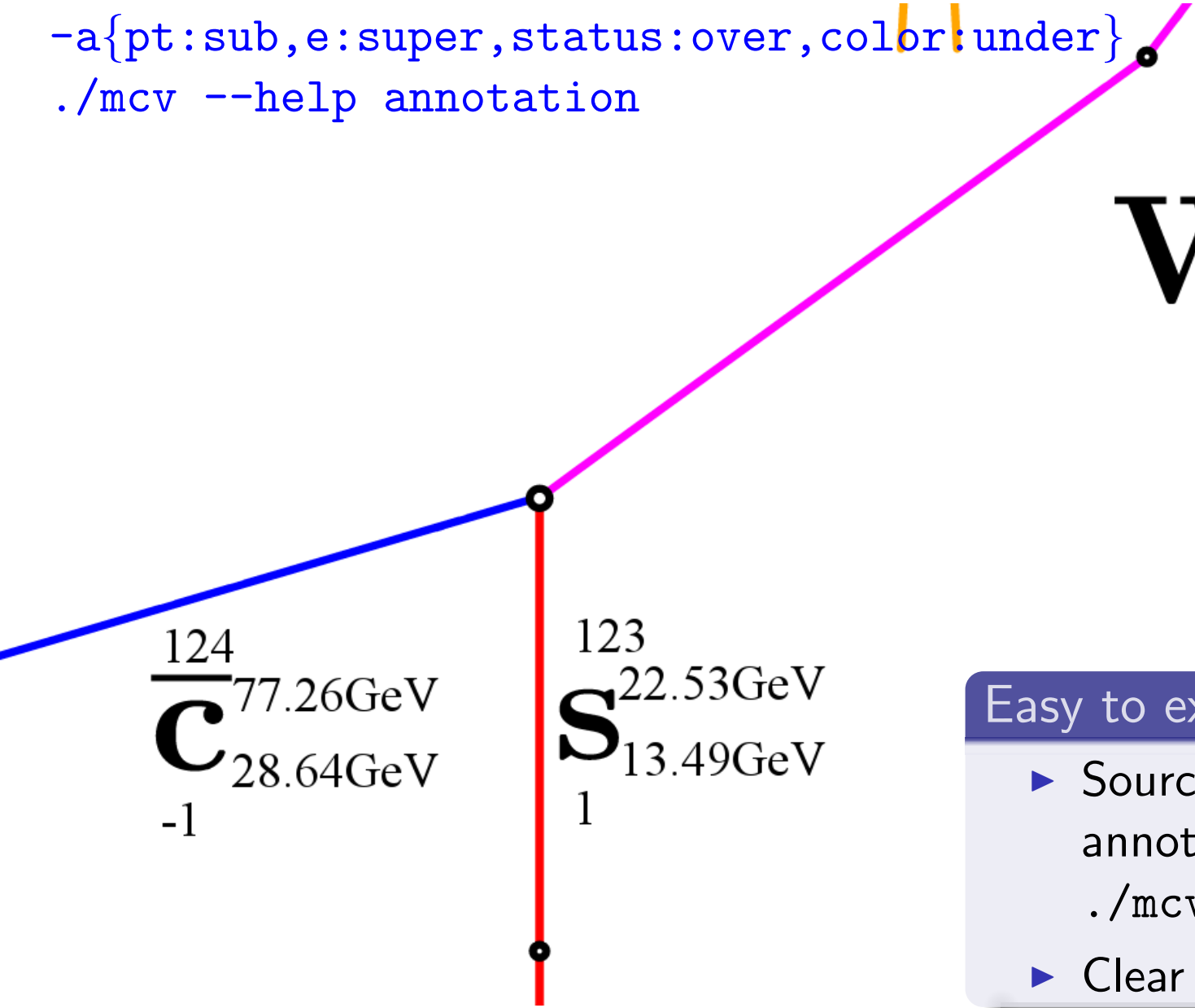
```
./mcv -sSimpleColors my.lhe && sensible-browser mcviz.svg
```



"Google Maps"-style interface - mousewheel zoom + click and drag

Examine status codes and momenta

```
-a{pt:sub,e:super,status:over,color:under}
./mcv --help annotation
```

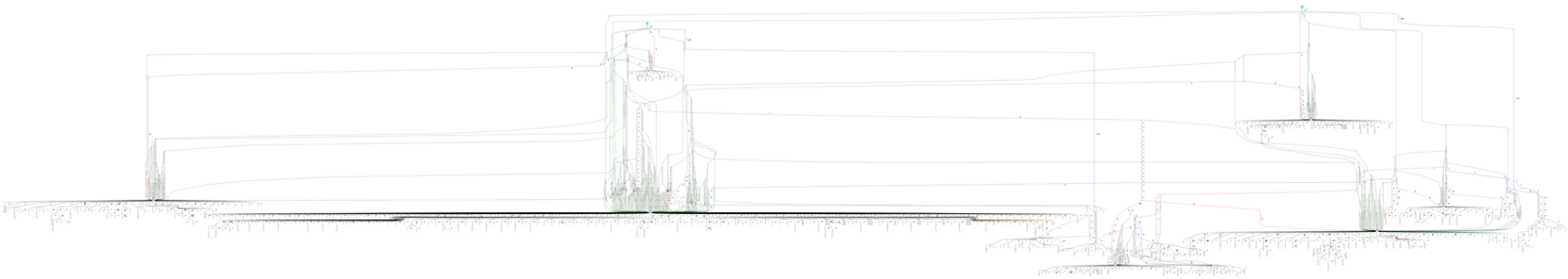


Easy to extend!

- Source file for existing annotations listed in `./mcv --help pt`
- Clear code structure

Tools: Summarize and Cut away Particles

Example: Pythia semileptonic $t\bar{t}$

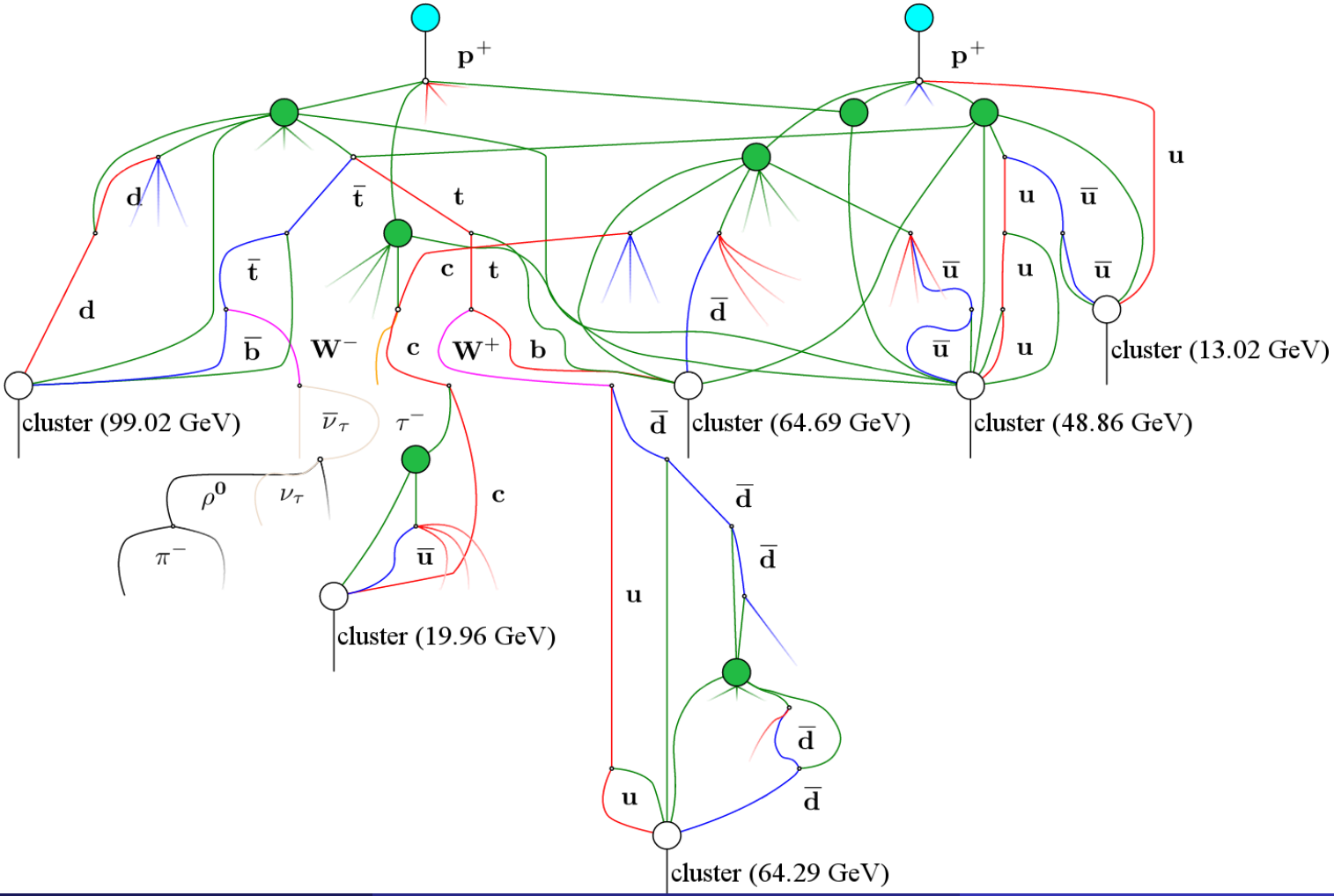


...2480 particles in the event record

Tools: Summarize and Cut away Particles

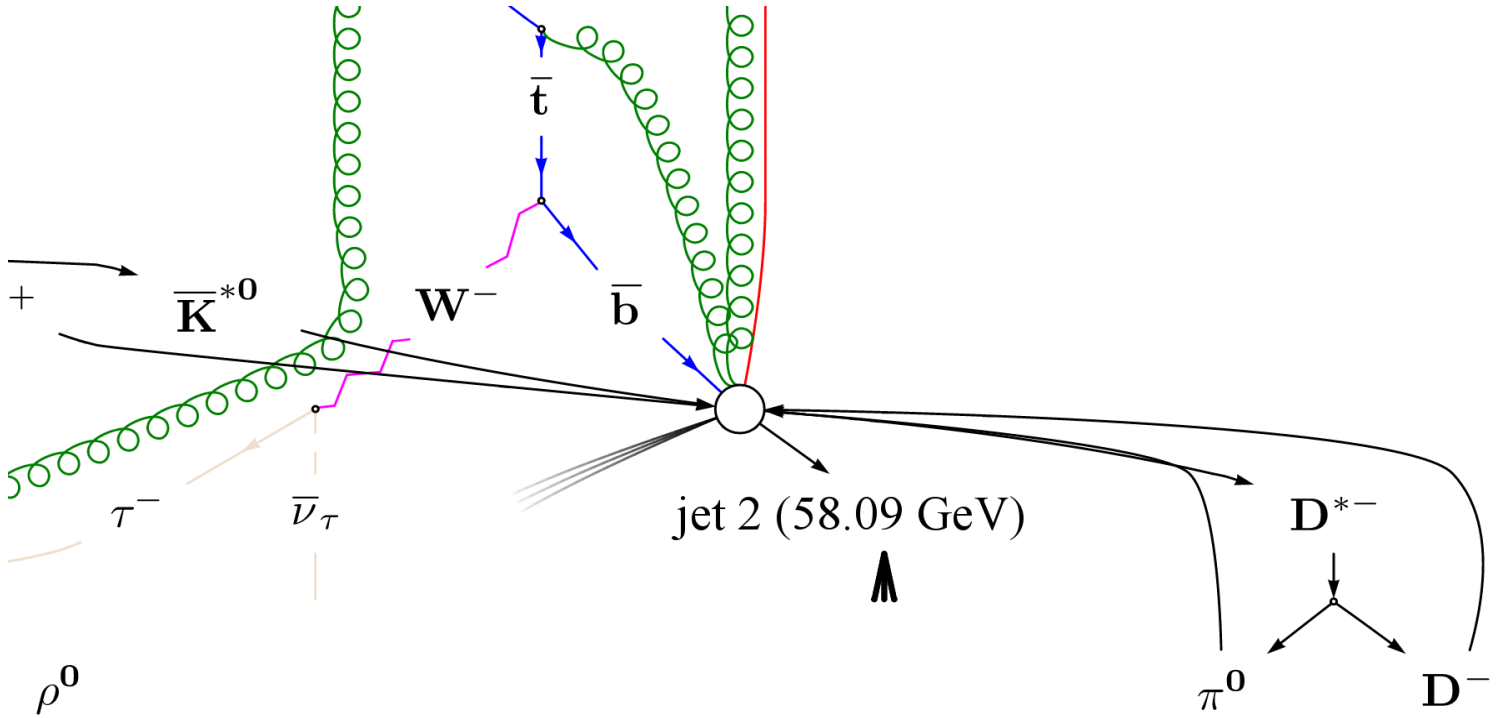
Same event with summarized hadronic clusters and $p_T > 10$GeV

`-tGluballs -tChainmail -tNoKinks -tNoLoops -tClusters -tCut:cut=10:param=pt`



Integrated Fastjet: Apply Jet Algorithms

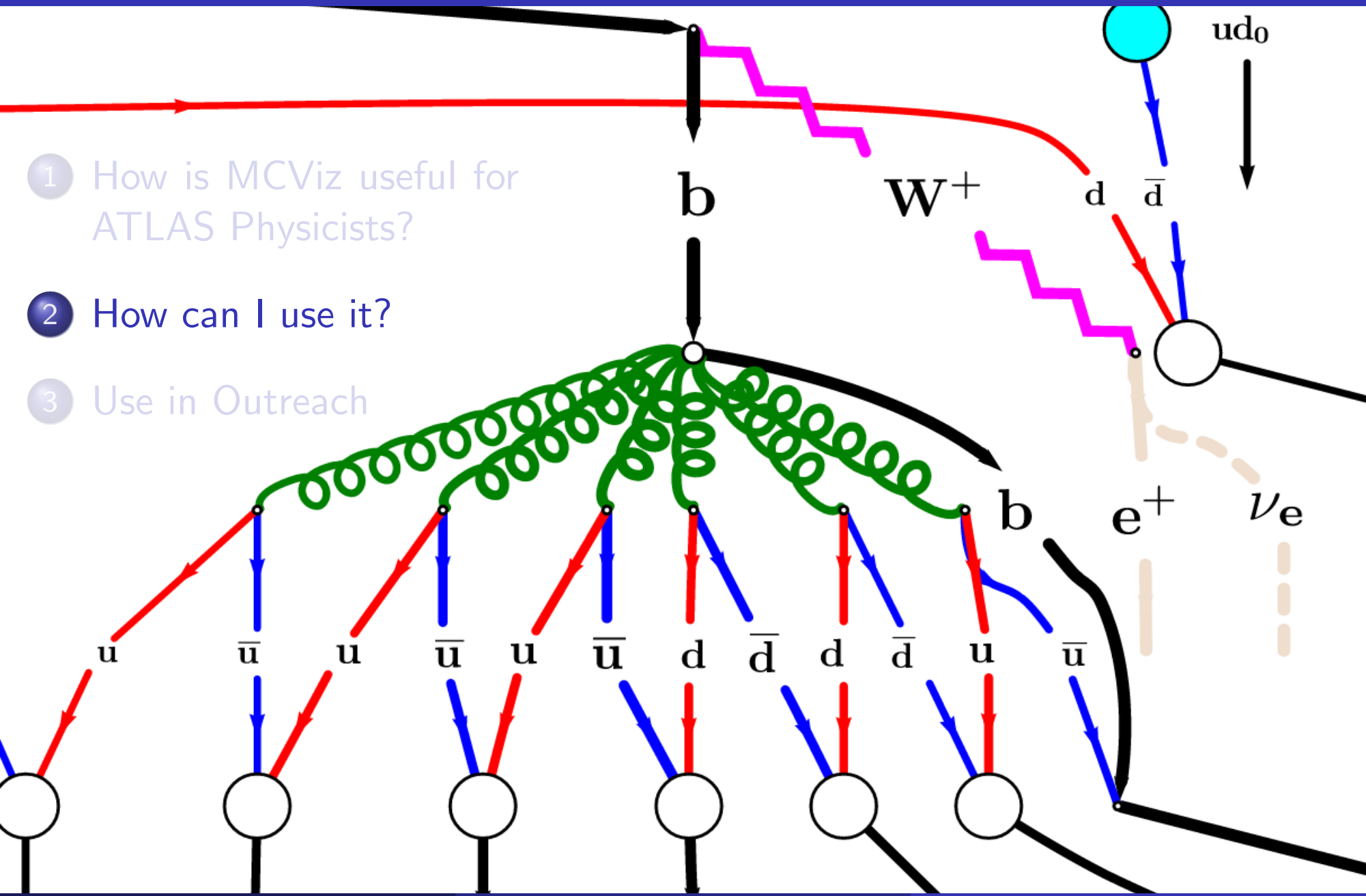
```
...  -lInlineLabels -sFancyLines -tJets -tCut:25:pt
```



`-tCut:25:pt -tJets` would first apply the cut and then fastjet - good for applying fiducial cuts!

(Jet display still needs some work, as you can see by the loop on the right-hand side)

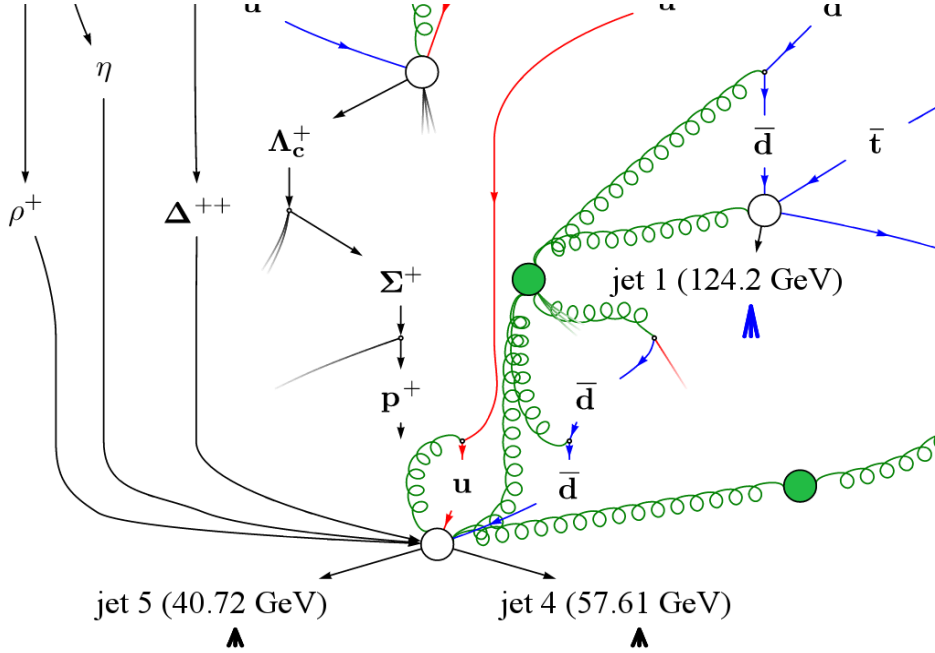
Outline



Using MCViz on ATLAS Data

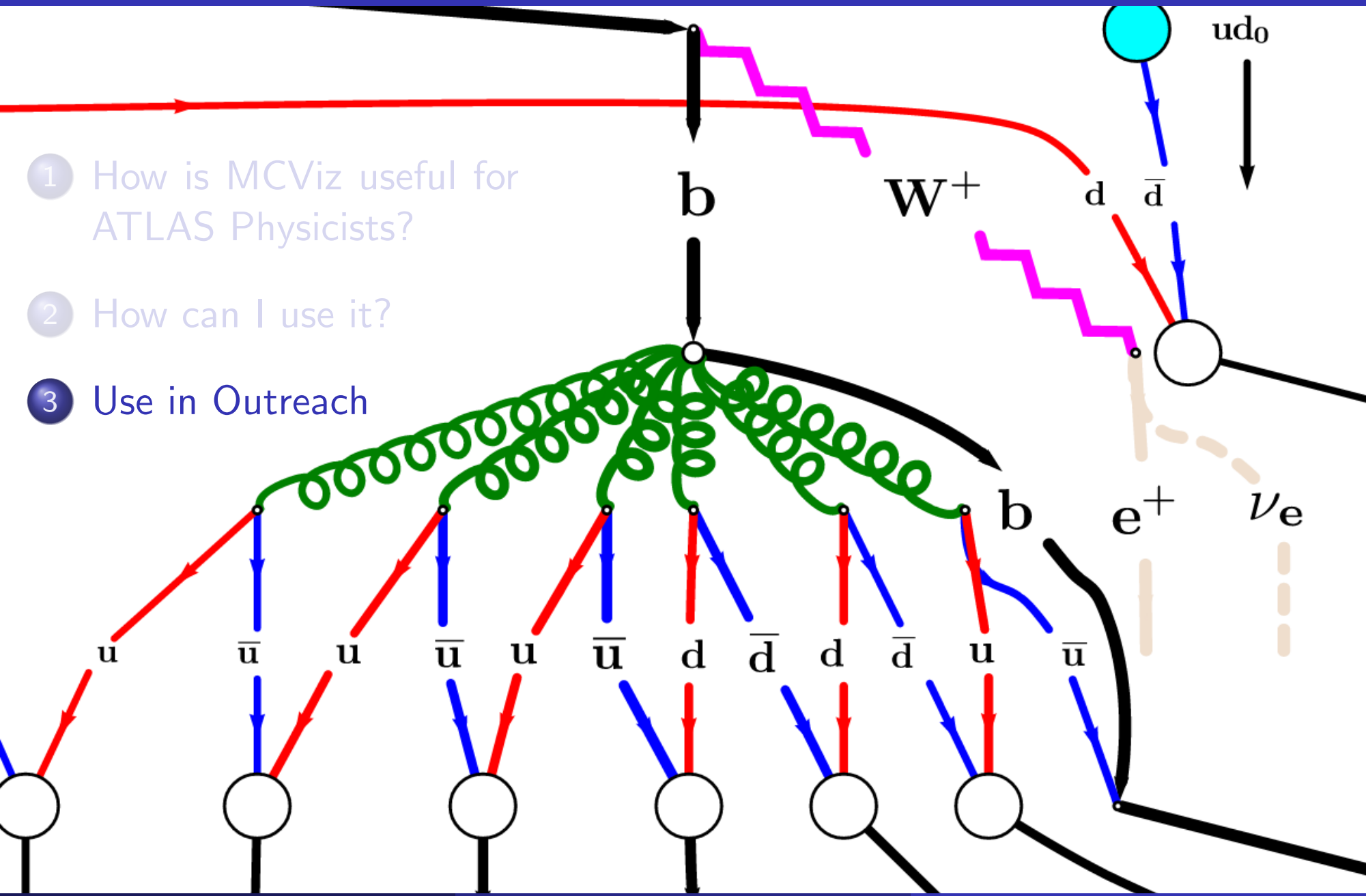
MCViz natively accepts

- LesHouches events (lhe)
- HEPMC files (hepmc)
- Pythia logfiles (log)
- (all also gzip/bzip2ed)
- Tested with Herwig, Pythia, Sherpa and MC@NLO.



- Unfortunately, ATLAS AODs seems to lack some necessary information (old status - maybe Andy has updates?)
- Fortunately, EVNT files are small and quick to `dq2-get`
- Small jobOption fragment to dump hepmc from EVNT is at `https://github.com/mcviz/mcviz/wiki/HepMC-from-Athena`

Outline



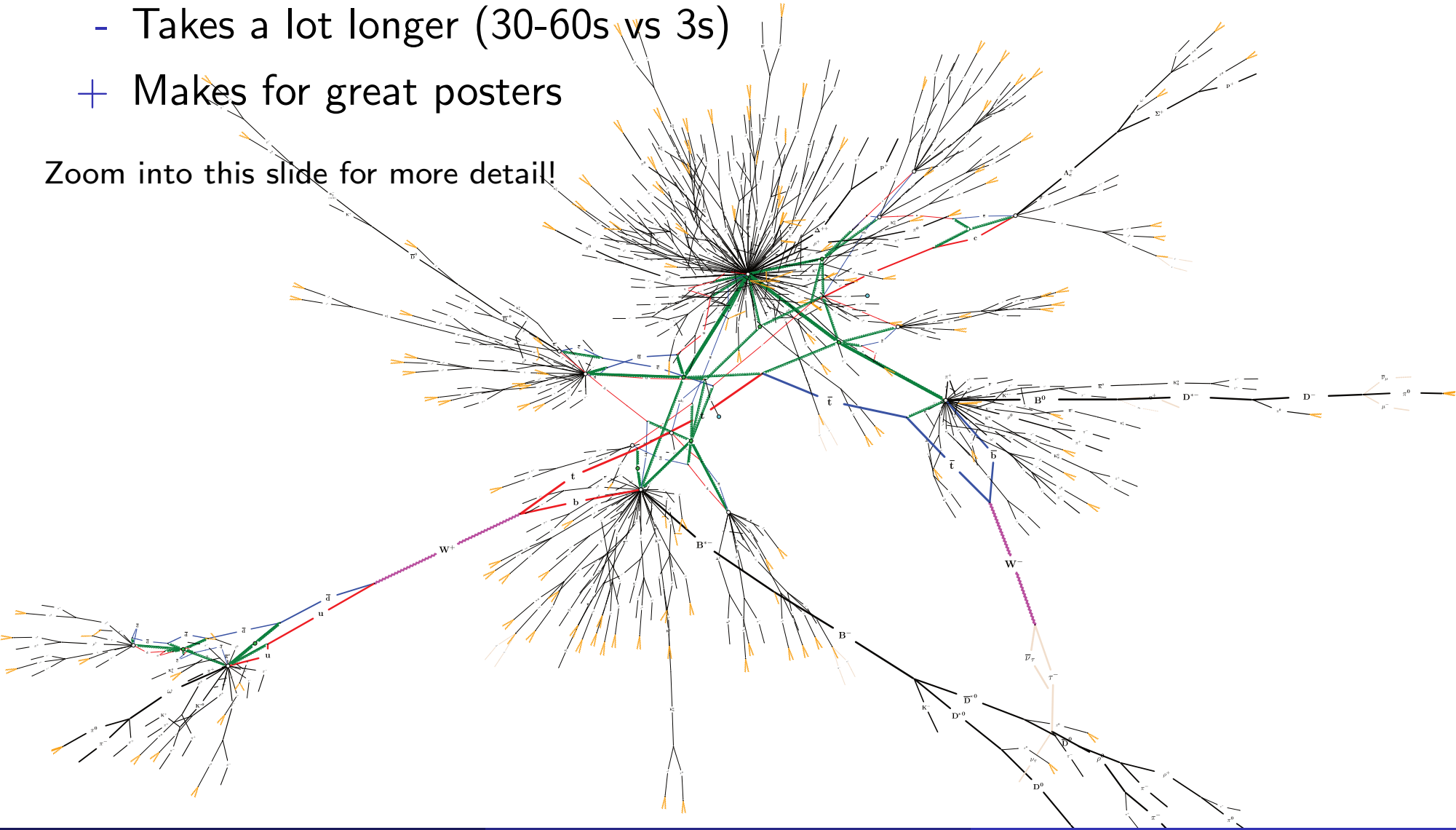
Free-form graph mode

- Use the graphviz "fdp" (force-directed graph) tool: `-e fdp`

- Takes a lot longer (30-60s vs 3s)

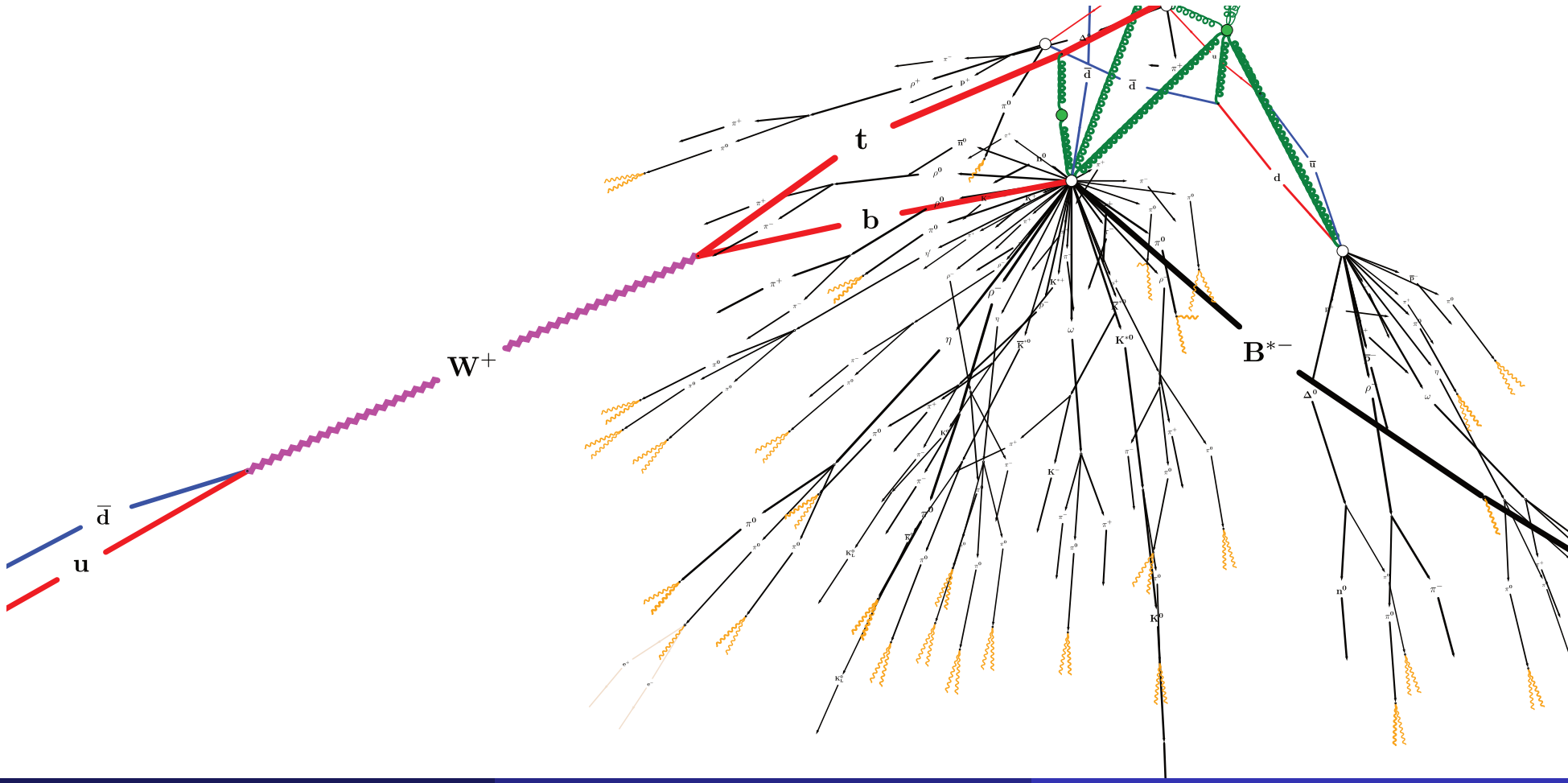
+ Makes for great posters

Zoom into this slide for more detail!



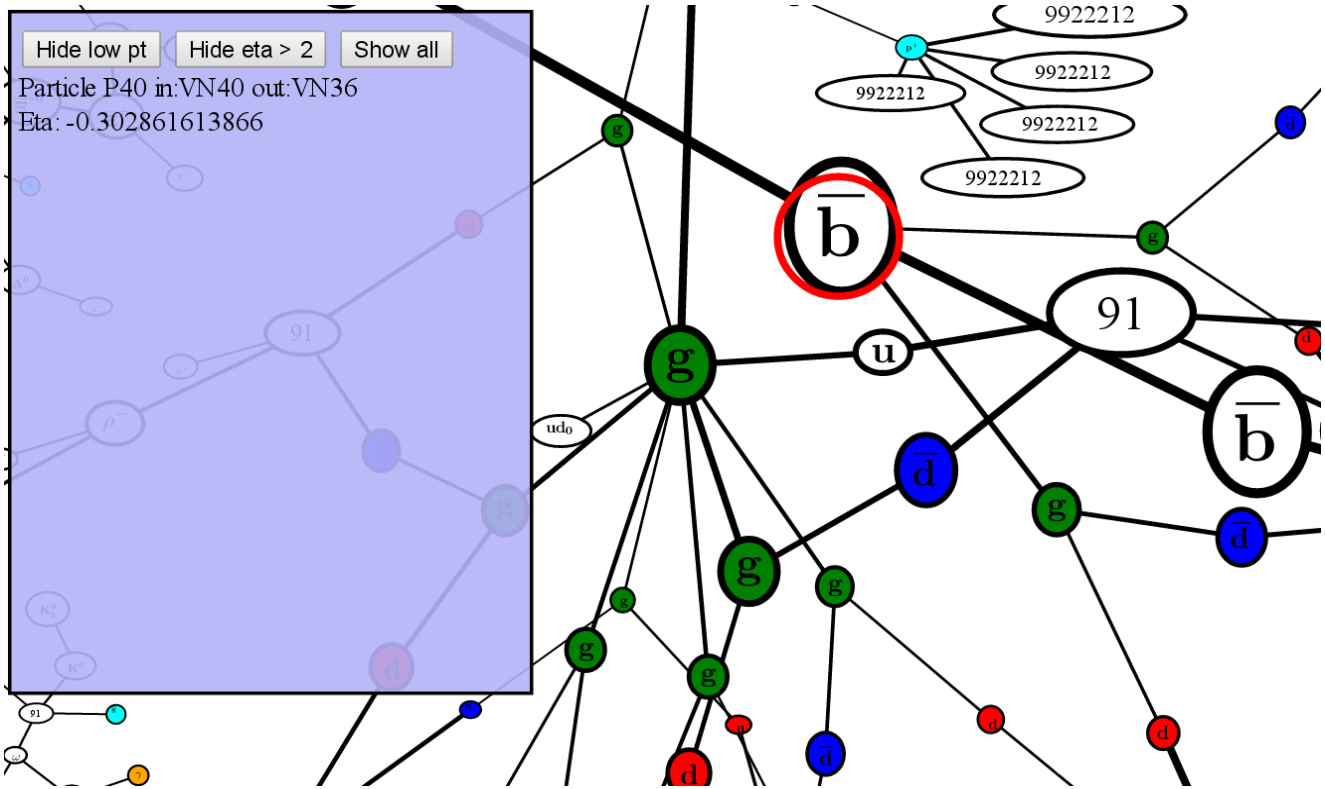
Fancy lines and p_T-dependent sizes

```
-efdp -lInlineLabels -sSimpleColors -sFancyLines
-sLineWidthPt -sLabelSizePt -tNoKinks -tGluballs
-tChainmail -tNoLoops
```



In development: Interactive browsing & 3d view

`-pwebnavisvg:mcviz.svg`

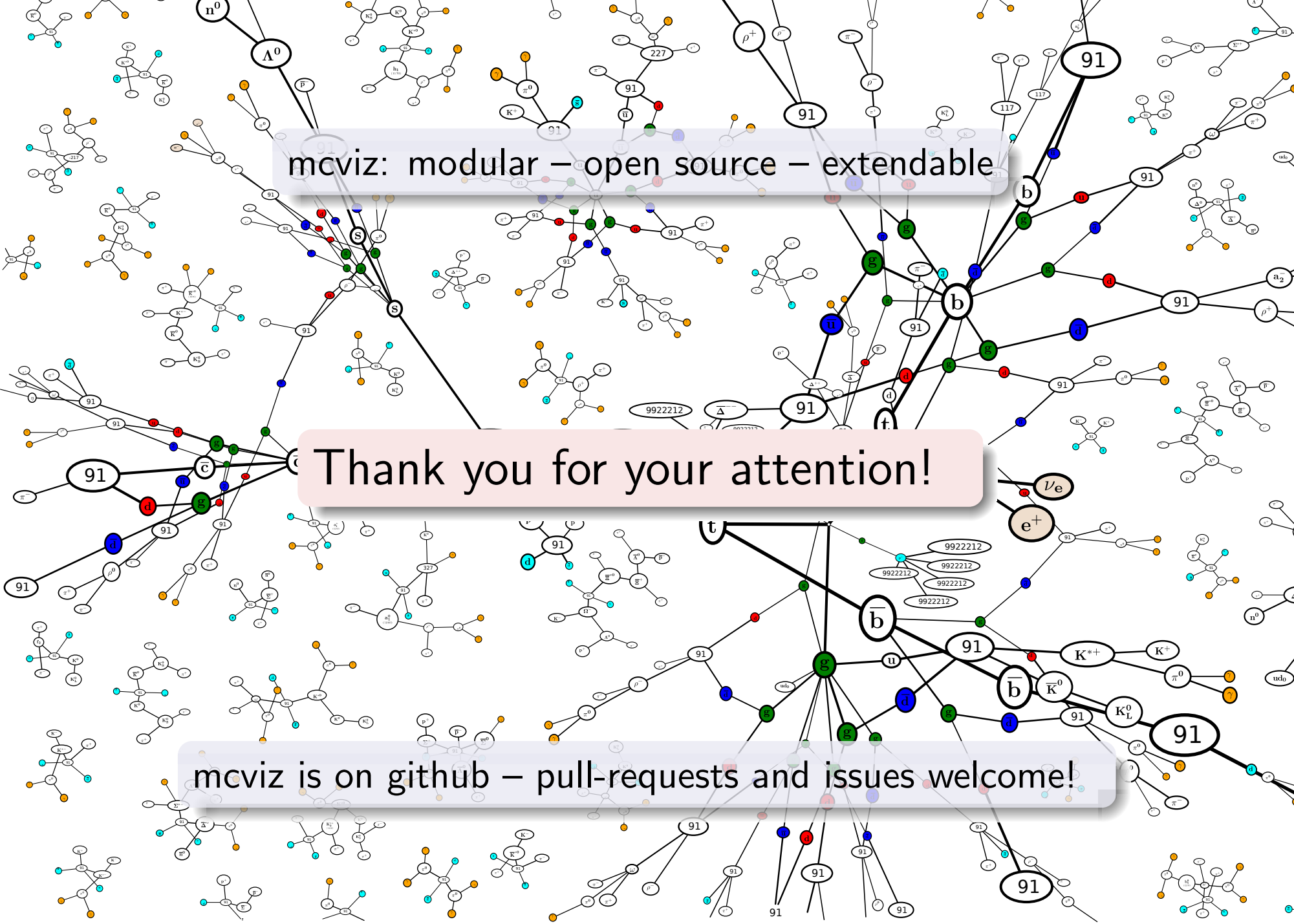


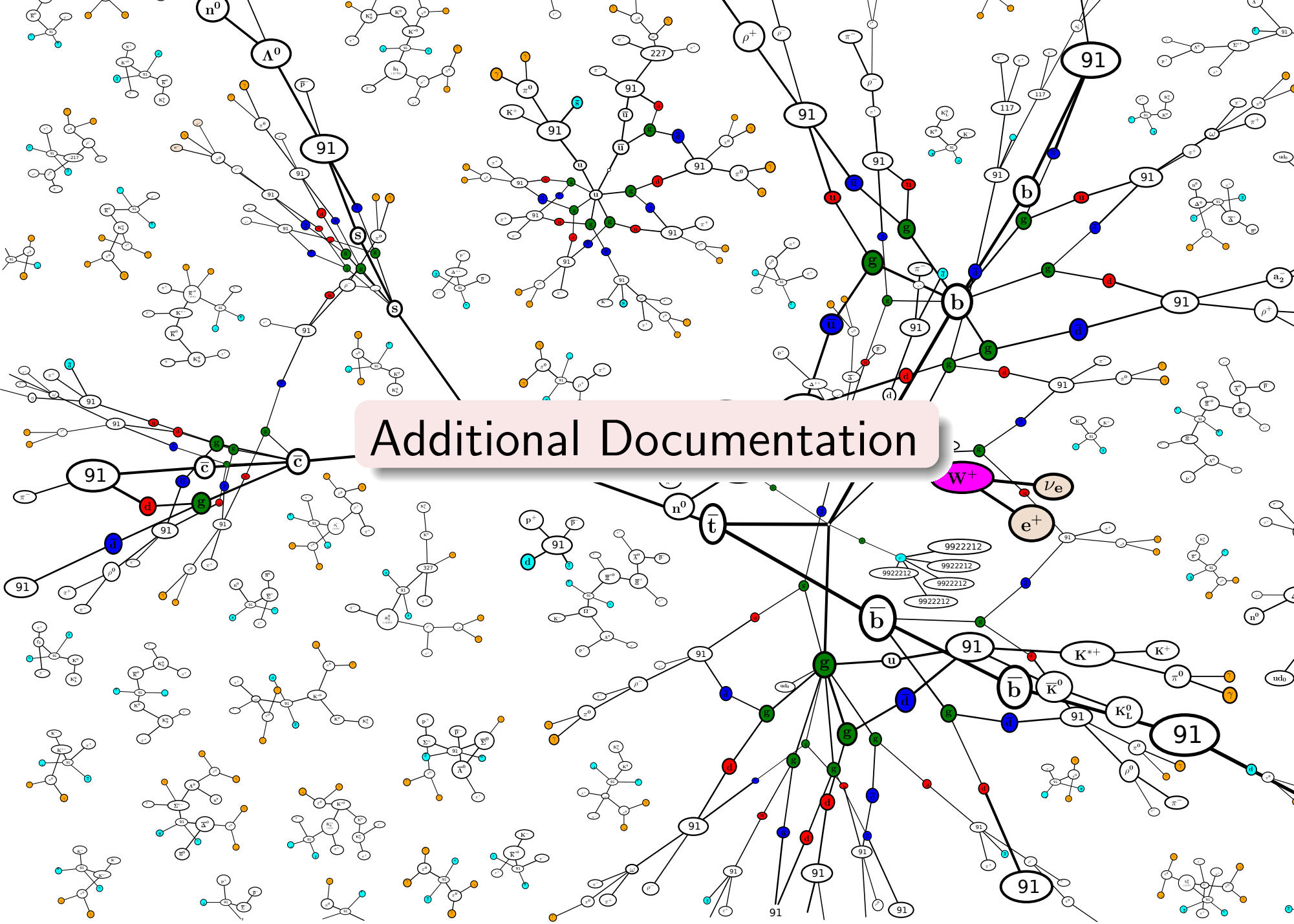
Try it live at `http://www.ebke.org/mcv2.svgz`!

Caveat: Not much information yet. Horribly slow with Firefox - use Chromium.

Experimental 3d view at

`http://pwaller.net/mcviz/viewer-momentum.html`


mcviz: modular – open source – extendable
Thank you for your attention!
mcviz is on github – pull-requests and issues welcome!


Additional Documentation

URLs and Setup

Useful Links:

www `http://mcviz.net`

dev `https://github.com/mcviz/mcviz`

wiki `https://github.com/mcviz/mcviz/wiki` (Examples)

Quick Start Summary:

- `git clone https://github.com/mcviz/mcviz.git`
- `cd mcviz`
- `./mcviz_bootstrap.py`
- `./mcviz_bootstrap_jets.py` # (setup fastjet - optional)
- `./mcv --demo mcviz.examples/mcviz/examples/inputs/pythia/LHC/out01`
- `sensible-browser mcviz.svg`

The MCViz Toolbox

MCViz accepts a list of tools to apply which produce the resulting image.

The implementation of these tools is typically quite short and reasonably readable, you can easily write your own!

Tool classes are run in the order:

- -t Transforms - modify the input event graph (i.e, apply cuts, etc)
- -a Annotations - annotate glyphs with additional particle information
- -l Layout - particle graph mapping onto the layout graph
- -s Style - line rendering (i.e, what colours thing are, curly lines?)
- -e Engine - physical positioning of the resulting edges/nodes
- -p Painter - rendering of the resulting graph (svg? custom renderer?)

Tool Chaining and Parameters

Multiple tools of the same class can be run - they stack against each other and order matters. Each tool takes the output state of the previous tool as input.

A sampling of the tools follows in the subsequent slides.

If in doubt, consult:

- `mcviz --help all | less`
- or our wiki at `https://github.com/mcviz/mcviz/wiki`

Some tools accept parameters separated by `:`, for example:

- `-edot:orientation=lr`

specifies that the graph should begin on the left and end on the right.

Tool class: Transforms -t

These change the physical structure of the graph. MCViz supports summarizing a number of vertices into one.

Examples:

- `-tClusters` summarizes everything below hadronization vertices
- `-tCut` can remove particle according to their physical properties
- `-tNoKinks` removes "particle $\rightarrow$ same particle" vertices
- `-tOnlyHard` attempts to remove everything in the graph not in the hard interaction (pythia only)
- `-tJets` runs fastjet to determine what the jets are
- `-tGluballs` summarizes all gluon self interaction

Tool classes: Annotations -a and Layout -l

Annotation: `-a`

Puts text describing the particle near its symbol. Examples:

- -a pt
- -a index:under

Layout:

- `-lFeynman` shows particles as lines, whereas `-lDual` shows them as vertices in the resulting display.
- `-lInlineLabels` interrupts each line to insert the particle's glyph as a node, and can produce a more visually pleasing result.
- `-lFixHad` causes the hadronization vertices to be positioned at the rank in the resulting graph. (sometimes, at least...)

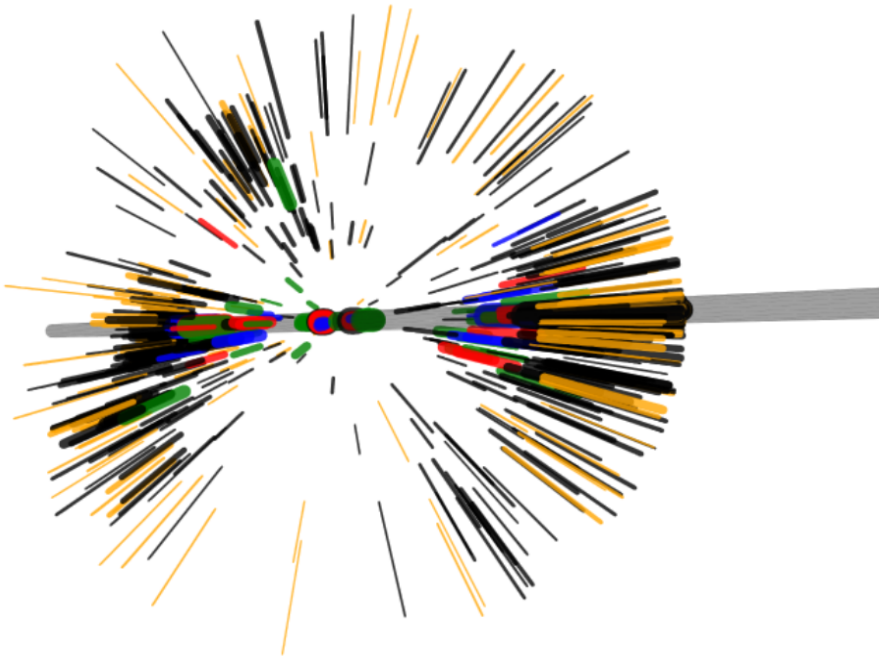
Tool class: Style -s

- `-sSimpleColors` makes quarks red, antiquarks blue, gluons green, photons yellow, leptons almond, bosons purple and neutral particles black.
- `-sQCDRainbow` gives each colour number a random hue, with the coloured particle being lighter and the anti-particle being darker. Makes it easy to see colour-cancellation.
- `-sLineWidthPt` shows line width proportion to the log p_T of the particle
- `-sFancyLines` gives gluons curly lines, photons sine waves, etc.

Tool class: Engine –e

The layout engine is the thing which takes the graph we want to draw and gives each edge a bezier curve and node a position. For this purpose we currently use graphviz, which comes in various flavours: dot, circo, fdp, neato, sfdp and neato.

It's possible to create an alternate layout engine which uses physical properties of the event to do layout instead of just the topological structure.



Experimental interactive 3d display

Tool class: Painter -p

The painter is responsible for taking the output of the engine and turning it into some form of physical image that you can look at. Currently, this means an SVG. Our SVGs come in different flavours so that we can supply navigation controls for the browser if they are desired.

navisvg is short for "navigable" as in, "you can navigate this google map". Currently, there is

- `-pdot` for raw graphviz output
- `-pnavisvg` the default - SVG with embedded click'n'drag javascript
- `-psvg` Only SVG without javascript
- `-pwebnavisvg` SVG with experimental javascript

For more detail come and take a look at the implementation, or shoot us a question at `mailto:dev@mcviz.net`!

Implementation

`https://github.com/mcviz/mcviz/tree/master/mcviz/tools`

Have Fun!