Using Zoom Technologies to Display HEP Plots and Talks

G. Watts
Box 351560
Department of Physics, University of Washington
Seattle, WA, 98195
USA

gwatts@uw.edu

Abstract. Particle physics conferences and experiments generate a huge number of plots and presentations. It is impossible to keep up. A typical conference (like CHEP) will have 100's of plots. A single analysis result from a major experiment will have almost 50 plots. Scanning a conference or sorting out what plots are new is almost a full time job. The advent of multi-core computing and advanced video cards means that we have more processor power available for visualization than any time in the past. This poster describes two related projects that take advantage of this to solve the viewing problem. The first, Collider Plots, has a backend that looks for new plots released by ATLAS, CMS, CDF, and DZERO and organizes them by date, by experiment, and by subgroup for easy viewing and sorting. It maintains links back to associated conference notes and web pages with full result information. The second project, Deep Conference, renders all the slides as a single large zoomable picture. In both cases, much like a web mapping program, details are revealed as you zoom in. In the case of Collider Plots the plots are stacked as histograms to give visual clues for the most recent updates and activity have occurred. Standard plug-in software for a browser allows a user to zoom in on a portion of the conference that looks interesting. As the user zooms further more and more details become visible, allowing the user to make a quick and cheap decision on whether to spend more time on a particular talk or series of plots. Both projects are available at http://deeptalk.phys.washington.edu. The poster discusses the implementation and use as well as cross platform performance and possible future directions.

1. Introduction
Experimental high energy particle physics produces an overwhelming amount of information. There are many conferences on different particle physics topics and the experiments are constantly releasing preliminary results, and publishing papers. It is impossible to track everything.

Worse many of the web sites that host conference slides or pre-release results from the experiments do not enable quick and easy browsing (or searching). For example, the process of examining a talk on an Indico conference site involves finding the agenda location, clicking on the talk link on the agenda, waiting for a PDF viewer to load up, looking at a few slides before deciding if the talk is worth it, and then repeating. When there are 100’s of talks this process is time consuming. Looking for the most recently released preliminary results is more difficult – no experiment follows any standard layout. Frequently the web page layout differs by sub-group within the experiment (the LHC experiments have gotten better at this). The projects discussed in this paper are attempts to address this.

The current state-of-the-art is quite good—for example an Indico agenda with PDF’s, or PPT’s, or their equivalent [1]. But that interface isn’t built for quick glances—it is built for deep and careful
examination, looking for something that the user already knows exists, for running a conference or meeting, and for ease of use for the speakers uploading the talks. There are other tools besides Indico, but Indico has gained broad adoption in the field and has an interface one can write code against. Preliminary results and plots are in much the same situation. The different experiments use different ways of releasing the results. The CMS experiment, for example, uses CDS. ATLAS uses custom generated web pages. CDF and DØ, two Tevatron experiments, also use custom webpages, some of them hand generated. And organization is not always intuitive.

The work described in this paper encompasses two projects connected by the underlying technology. One of them, DeepTalk, has been around and evolving for a number of years. DeepTalk takes an Indico agenda and lays out all the slides from the conference in a large grid and allows seamless zooming in and out to view all the talks from the conference quickly. There is also a personal version of the DeepZoom rendering code. The second project, Collider Plots, scans experimental websites for plots and catalogs them and tags them with searchable metadata. The user interface displays them in an interactive large data browser designed to graphically explore large numbers of items.

Both projects grew out of ideas that were first developed at Microsoft – the DeepZoom technology and the PivotViewer.

Though the projects are technically related, they are quite different in their use. The first half of this paper, Section 1., describes the DeepTalk project, and the second half describes the Collider Plot Explorer. Each section discusses the project from the user point of view and also describes the back-end implementation.

2. The DeepTalk Project

The layout DeepTalk generates is hierarchical, as shown in Figure 1. This is the CHEP 2012 conference, rendered about month after the end of the conference. Imagine all the title slides of all the talks are expanded to a very large size and laid out on a large conference hall floor. Then, under each title slide the rest of the presentation slides are placed in a much smaller size. The talks are then grouped by conference track or section. Finally, start with a camera up very high and then steadily zoom in. This is the basic idea behind DeepTalk. The section headings and talk title pages (or posters) can easily be seen when zoomed out. Under each talk title page the slides are all arranged. As the viewer zooms in they are treated to more and more detail (sections, title pages, slides, and the contents of the slide).
The DeepTalk project and its conference renderings are currently served from a web server located in the Physics Department at the University of Washington. It maintains a list of the rendered talks and also has a URL design to allow for easy external linking to a rendered meeting. The website can be found at http://deeptalk.phys.washington.edu.

The first version of this project was a few weekends of work. Actually make it work well, of course, required a lot more effort (the devil is in the details). CHEP 2009 described this work, and this section of the paper will address only a basic outline of how it works, instead focusing on improvements and changes [2].

2.1. What is DeepZoom?
The DeepZoom algorithm is fairly common. Most map web sites use some form of it. As the user zooms in on a map of the United States, the web browser progressively loads more detailed images of the area shown in the browser. Zoomed out one will see only major roads and land features. The closer you get the more you see of city streets or even houses if the satellite overlay is enabled. At full resolution a full high resolution image of the United States would likely be 10’s of terabytes in size. However, the browser window shows only a very small bit – and so the technology behind the viewer only sends that small segment of the image to the browser. This makes quick performance possible – even to mobile devices – and also greatly saves on bandwidth. The disadvantage is the view with reduced detail (layer) must be pre-rendered and stored on the server, increasing the size of the stored data.
Instead of a map of the United States, think of all the slides of a conference laid out on a very large floor. For DeepTalk, that is the high resolution image. A decent length talk rendered this way can require almost 20 levels when rendered at 200 DPI. For DeepTalk, each tile is configured to be 256x256 pixels. When the jpeg is written at 95% quality level the sizes vary between 4 and 30 KB each, depending on complexity of the portion of the slide image in each tile. Small files like this can be quickly downloaded over the internet.

Once the images arrive at the web browser, Silverlight [3] stitches them back together and scales them correctly to appear in the user’s web browser. Silverlight also deals with zooming the images between tile steps which, unlike most mapping programs on the net, makes the transitions appear quite smooth, though it takes a great deal of CPU and GPU resources.

2.2. The Rendering Engine

The basic workflow of the rendering backend is shown in Figure 2. This currently runs on a desktop computer at the University of Washington, separate from the web server. While this toolset will render most the formats that most users use to create their slides it could easily be expanded in the future. As to be expected, the tools are a mixture of code written by the author and various open-source tools easily available on the net.

![Figure 2: The workflow to render a conference or meeting. Once an XML version of an Indico agenda is downloaded from the conference web site, each talk is individually downloaded and converted to JPEG images. With the images in hand, a layout library arranges them and coordinates their rendering into the DeepZoom tiles which can be displayed.](image)

The workflow is fairly straight forward. The rendering backend starts with the URL of an Indico agenda. Modern versions of Indico can render the agenda in an XML format which the backend parsers. Besides meta-data for each talk, URL’s for the slides from each talk are extracted from the XML.

The talk slide file downloads are carefully coordinated so that the conference website isn’t hit with a large number of download requests at once. The system prefers PowerPoint or OpenOffice formats over PDF, and PDF over Postscript. The main reason is PowerPoint 2010 does a better job of turning a PowerPoint or OpenOffice file into a PDF file than tools that do this automatically do (definitely better than the default Indico PPTX to PDF converters). And postscript is the least favored due to problems with margins and interpreting page orientation. Indico is inconsistent in the way it serves files – for example the MIME type isn’t set at all for .pttx files, which requires some working around if one desires the downloading code to be relatively secure. There are other variations between different Indico installation sites which must also be worked around.

Once the files have been downloaded they have to be converted to JPEG files. PowerPoint and OpenOffice files are converted in two steps. The PowerPoint files are converted to PDF’s using PowerPoint 2010. This conversion process isn’t error free, unfortunately, as some unknown picture
formats (mostly PowerPoint files generated on Apple computers) will cause PowerPoint to prematurely abort the conversion. When they do happen they are fairly easy to recover from – however the whole talk will not show up in the final rendering.

The *GhostScript (GS)* program is used to convert PDF and Postscript (ps) files to JPEG images. This tool has been the most problematic in the tool chain. There are numerous PDF files that cause GS to crash – that are rendered without error (or with a single warning) by Adobe. Crashes are easily detected and can be easily cleaned up. What is more insidious is a certain class of files that cause GS to enter an infinite loop. The rendering chain automatically detects this and kills off GS if it doesn’t seem to be making progress. Recent versions of GS have substantially improved its ability to deal with these malformed files.

Once all the talks have been converted to JPEG images the conference layout can be generated. A simple image library written by the author gives the basic tools and coordinate transforms to do this easily.

Rendering to tiles is done by a set of tools released by Microsoft, called DeepZoom Composer. Each talk is rendered individually so that future updates to the conference only have to render the new talks. Further, the rendering takes up a good deal of memory – it isn’t possible to render more than about 50 slides in a single job with the present tools.

A website on an old single-core machine serves the talks and the associated data. The talk data itself is static. The webservice doubles as the backend coordinating the rendering of talks and maintaining the rendering database of rendered talks. This database also drives the home page of the DeepTalk site, displaying the most recent 20 rendered talks.

A web service manages the data – there are TCP and HTTP endpoints exposed using the SOAP protocol. There is no reason JSON and RSS couldn’t be exposed¹. All code was written in the C# language.

The rendering database tracks when each Indico conference meeting is occurring and will repeatedly re-render the conference as it approaches. During the actual conference it will trigger a rendering once every 4 hours. After that the updates are done once per day slowly, decaying away to no further renderings 6 months after the conference. The system is designed to cache the renderings of each talk – if there has been no change in the uploaded talk then a re-rendering is not performed.

The Silverlight control that displays the talks on the user’s computer is fairly basic. The talk is displayed as a very large image that one can zoom in. The titles for the talks and the conference itself are active: clicking on them will take you to the conference Indico site or to the file online that was rendered. This allows the user to get to the source material. Once one is looking at a talk it is possible to use the left and right arrow keys to navigate the talk itself. The up arrow will take the user back to the section and the conference and allow some navigation there; however, the algorithm for navigation isn’t well tuned to scan conference sections, just talks.

### 2.3. Personal DeepTalk

An easy-to-install application is available from the DeepTalk CodePlex web site, [http://deeptalks.codeplex.com](http://deeptalks.codeplex.com). This will install the complete tool chain, except for GS and MS Office, on a Windows desktop. When run it will request a URL or location on disk and proceed to render that to a DeepTalk image. An Indico agenda, or a directory structure that contains PDF files (papers, for example), can be given to it. The core rendering code is identical to what is described in Section 2.2. Using this standalone program one can render conferences or local directory hierarchies that have some personal significance.

The rendering architecture is just a series of plug-ins. For this project a plug-in that parses the modern PowerPoint file format was added. The rendering code finds all shapes in a PowerPoint file and scans the shape’s alternate text for lines that start with “*render:*”. The rest of the line is then
rendered and scaled to fit inside the shape. For a simple example of something that the author used to help prepare for a workshop see Figure 3.

The original plan for this was twofold. First, one could create a topical poster. The poster could contain a broad high level explanation for all parts of some project or physics analysis. A viewer could then zoom in to find details – papers, talks, etc., that were relevant to the topic. Secondly, it was envisioned as being useful for conference preparation. In this case on slides showing some topic a small, invisible during-the-talk shape could be added that contained the render command. Later, when the talk was rendered, full details of an analysis could be embedded in the talk.

Figure 3: A PowerPoint file containing render commands embedded in the shapes. The left image is the raw PowerPoint file, and the right image is the post-rendering result. Note the full Indico agendas that are rendered in the upper right hand corner. The other items are files in a local directory structure that have been rendered.

2.4. Comments and Experiences

This system is a resource hog—it requires high bandwidth network, lots of memory, and lots of disk space. Most of the tools worked well for the project, though some of them are more complex than they need to be for this project (i.e. the web service infrastructure). It was very nice to be able to program end-to-end in a real programming language (C#) - from the server (CLR) to the web browser (Silverlight). That said the backend code has been stable for a number of years and at least one operating system upgrade with almost no changes. At CHEP 2009 there were serious stability problems. At this point the author needs to rarely check on the backend – it just runs.

There are a lot of small issues that need to be fixed—the long tail. The variety and quality of PDF files out there, or the large number of ways that Indico can place a talk in XML were some of the more frustrating things that had to be coded around.

The personal rendering tool has been very convenient. However, it has not been as successful as wished because of the length of time to render items. It takes about 5 second to put an Indico URL into a PowerPoint shape. But it can take hours to render it. A single talk can be done fairly quickly – a few minutes, but big conferences can take hours. Part of this is due to the single threaded nature of the rendering process.

I observe people using the system steadily, though at a small rate; so in that sense it is a success. It isn’t clear if people just explore the system, commenting that it is “neat” and then never return, however.

The code is open-source—and available on the open source CodePlex site: http://deeptalks.codeplex.com.

2.5. Future Directions

A number of future directions for this component of the project are possible. Primary among them is moving away from the Silverlight dependency. This would open up use to all users and, further, get past the folks that refuse to install any Microsoft software on their platform. JavaScript and the modern GPU graphics in browsers should make it possible to do this. The author does not particularly
enjoy JavaScript or dealing with the cross platform incompatibility issues, however. However, it is either this or understand how to write apps for iOS, Android, and Windows.

The website currently displays only the 20 most recent talks – though there is a big archive now collected (about 600 GB of conference data). Search functionality is needed to access these past conferences. It would also be very nice if one could combine talks across different talks – say search for “Higgs” and see talks from all conferences with that in their title.

3. Collider Plot Explorer

The Collider Plot Explorer is based under the same underlying DeepZoom technology as DeepTalk. However, a fairly sophisticated Silverlight control to explore large datasets visually has been layered on top. Figure 4 shows the main window, in histogram mode, binned by date.

![Figure 4: The main window of the Collider Plot Explorer. It is displaying the most recent 4 months in a histogram (note the histogram mode selected in the upper right hand corner). The Moriond bump can be clearly seen. The pane on the left allows the user to window the data by selecting date, note title, experiment, group, or published vs. preliminary release data.](image)

The project can be cleanly split into two parts – the user interface and the backend. The user interface is mostly work that has come out of Microsoft Research’s work on visualizing large datasets. The backend is work done mostly by the author.

3.1. The User Interface and Using The Explorer

The tool can be downloaded from [http://deeptalk.phys.washington.edu/](http://deeptalk.phys.washington.edu/). It can be run in a web browser, or by right-clicking it can be installed and as a standalone program. The latter is recommended because it then becomes possible to click on links and follow them to web pages and image files.

A large amount of data must be transferred around in order to look at all the plots. A good CPU/GPU and a decent internet connection are requirements. In order to keep these data requirements under control the user can grossly limit the amount of data being looked at using the strip on the right hand side of the window. The default is *Recent* which corresponds to the last 4 months’ worth of plots. A particular year can be selected or *All* which is all data (but the user must be ready to wait for the loading to occur in that case!). Performance is not, also, that great when looking at all plots. The
database currently has about 30,000 plots. The last 4 months corresponds to just over 5,000 plots. Performance starts to degrade significantly after about 20,000 plots.

The top bar contains the data of the last scan, along with some general display options. A grid display and a histogram display are possible. The data is always sorted by some key – date, note title, experiment, or group. Group refers to the physics group or detector group that produced the plots (for how the plots are classified, see Section 3.2).

The pane on the right hand side of the window allows one to select the plots displayed. For example, starting from the display in Figure 4 one could select the **Experiment** category and select **ATLAS**. In that case, all non-ATLAS plots would be removed from the display. It is also possible to click on one of the histogram bins and drill down deeper. This is particularly useful when looking at the plots displayed in bins by date. Clicking on a Month will give you the plots binned by the day of the month.

It is also possible to click on a plot itself. The display zooms in on the plot close enough to for a full view, and also gives a good deal of extra meta-data for the plot. Figure 5 shows an example of this. The data listed is a figure caption (if that was available), direct links to any plot files (eps and png in this case), the date of the plot, the title of the note or webpage that contains the plot, the experiment, a link to the page where this note was found, the group, and, finally, the note type. The graphics files and the note page are active links. If the viewer is installed those links can be directly clicked on and viewed. This is ideal for getting the original high resolution image.

![Figure 5: The display zoomed in on a particular plot, with the metadata for the plot shown on the right.](image)

### 3.2. The Backend

Almost all development work for this project took place in the backend. The code is all open source and can be found on CodePlex: [http://dzpivotconstructor.codeplex.com/](http://dzpivotconstructor.codeplex.com/). The backend, architecturally, is very simple. It scans a collection of web sites and extracts plots and metadata for the plots from them. It then renders the plots using the same DeepZoom rendering pipeline as the DeepTalk project (indeed, the libraries are shared).

The web site scanner is essentially an intelligent web crawler built on a plug-in system. The webpages used by the various experiments are fairly uniform in format within a single experiment, or at least within a single physics group in a single experiment. They do undergo major revisions, but rarely. As a result a single plug-in is written for each style of webpage. The plug-ins know how to
identify the web pages they can parse. This system has proved to be very easy to write, debug, and test. For testing, a copy of the interesting web pages are downloaded and stored as test data. The interface to the plug-ins was designed to be able to accept this locally cached data. This makes for both fast and stable tests. If a web page is found that violates some assumption of the code, it is simply added as a new test case. Given how complex some of the HTML parsing algorithms can be, this has proved to be a huge boon. One is sure that the old format of the web page can still be parsed when a new type of web page is added because the old format remains a test case.

An experiment’s web pages are arranged in a hierarchy. Most experiments start with a single landing page for all public results. From there they link down to results from the Higgs group, from the Exotics group, from the tracking group, etc. The web crawler will tag all plots found while crawling the links under the Higgs group with Higgs. This is how the Group tag is associated with each plot. There are cases where a joint analysis, like a search for a t’, might be found in two locations (top and Exotics in the case of the t’). As long as there is only one copy on the website, this will be correctly recognized and the plot will be tagged as belonging to two groups.

The backend runs as a small task on the author’s desktop machine. It runs once every other night. Each run takes close to 8 hours. The code knows not to re-download a plot it already has, but it does download every single physics result web page: there is no caching mechanism for those currently. Only public plots are scanned – the backend scanner knows no passwords or any other secret information to give it access to secure plots.

There are special blocks in place to prevent the back-end visiting some sites. For a while the main physics pre-print site, arXiv, had blocked the machine the crawler was running on. Further, some experiments put plots and papers next to each other. When the crawler detects this it ignores the papers. Unfortunately, it doesn’t remember this fact run-to-run.

3.3. Future Directions
The usage of this tool is fairly good right now. The main features that need to be worked on are improvements of the backend – the speed it crawls the web, knowing that certain pages won’t get frequent updates (and thus can be checked infrequently), etc.

It would also be nice to rewrite the user interface in JavaScript. However, that is a huge job. A large number of Apple owners refuse to install Silverlight because it is from Microsoft and this limits the tools use (in particle physics).

Finally a RSS feed of recently released results, similar to the new feed that comes from arXiv, has been mentioned by several people as being an interesting addition to this project.

3.4. Comments and Experiences
This tool has been most heavily used by the author and others to look to see what the various experiments have recently released in a few clicks. The second use is finding results in preparation for a review talk. Finding all the recent results released when writing an Exotics talk, for example. For this use case the tool is fairly efficient.

The way the UI splits the things its views isn’t quite optimal, however it is very easy to work around. For example, when delving into a month it first displays the first and second half of a month rather than going straight to days or perhaps weeks. On a large monitor this doesn’t necessarily make sense.

4. Conclusions
Two tools to aid viewing particle physics results built on high speed graphics (the GPU) and many sources have been described. One, DeepTalk, is tuned for quickly viewing all the slides from a conference quickly. It includes a personal version which also has the ability to embed extra information in PowerPoint slides. The second project, Collider Plot Explorer, sources plots from the CDF and DZERO experiments at the Tevatron and the ATLAS and CMS experiments at the LHC and presents an interface designed to view and explore all of those plots at once.
References
[1] Indico Agenda System, http://cdsware.cern.ch/indico/
[2] DeepTalk: A Complete Conference in a Picture, G. Watts, CHEP 2009 proceedings
[3] Silverlight, http://www.microsoft.com/silverlight/