Course C
The Powerdot Presentation Class

The use of LaTeX to make PowerPoint-style presentations is relatively new technology. Even though the idea is only a few years old, people have already written many different document classes for this purpose. The most popular ones are the `beamer`, `(HA)prosper`, and `powerdot` classes. I will be explaining the use of the `powerdot` class today, mainly because it's the one I am most familiar with. If you wish to use another such package you should be in fine shape, since they all work basically the same way.

How Powerdot Works
Powerdot creates a pdf file that "acts like" a Microsoft PowerPoint slide show. Just as in PowerPoint, you have slides, overlays, transition effects, etc. You click through the slides in exactly the same way.

Powerdot uses some PostScript features to make the final product. Because of this, the normal compilation routine will not work: you cannot use a dvi viewer to get an accurate depiction of the appearance of the final product. Therefore, Powerdot files must be compiled by the following steps:

- Compile as an ordinary LaTeX file (maybe twice)
- Convert the dvi to a PostScript file in the *landscape orientation* (WinEdt people: just tap the dvips button, taking the landscape option)
- Convert the ps file to a pdf file (WinEdt people: just tap the pspdf button).

Helpful Hint: Acrobat will complain if LaTeX tries to copy over a currently-open file. Therefore you should close the original pdf file before attempting to re-compile any updates.

If all goes well, you now have a pdf file consisting of your slides. Give your talk in Adobe Acrobat using the full-screen mode (shortcut: `ctrl+L`). The `esc` key will revert back to the normal view.

Of course, to do any of this you actually need to acquire the Powerdot package. I think the most recent version of MiKTeX knows about Powerdot, so you can easily acquire it through the MiKTeX package manager. Otherwise, you'll have to obtain it through `www.ctan.org`.

Powerdot Settings
The toughest part of all of this is setting up the style of your slides. I have done the hard work for you, supplying a basic annotated template that you are free to modify. You should refer to the Powerdot manual for more information on your options.

The important point is that some options occur in the `\documentclass` declaration, while others appear in the `\pdsetup` command. There is actually a good reason for this. At any rate, I have already inserted the most common options in the template file: just comment out the options you don't want.

Making a Basic Slide
To make a new slide, start with `\begin{slide}{Your Slide Title}`. The slide title is *not* optional! Typeset as you normally would, ending your slide with `\end{slide}`.

Performing a basic overlay is also easy: just code `\pause` where you want the break to occur. So for example,

```
\begin{slide}{My Title}
Here is the first line on my slide.
\pause
And now here's the next.
\end{slide}
```

would create a slide with "Here is the first line on my slide." and nothing else. Upon the next click of the mouse, the line "And now here's the next." would appear on the same slide. As usual, you have to specify any vertical space, forced line– or pagebreaks, etc.

Fancier Overlays
Often, you'll have a bulleted list of facts that you want to appear not all at once, but line-by-line. There are several ways of doing this. The easiest way is simply to code a pause between the items, as in

```
\begin{itemize}
\item  My first item \pause
\item  My second item
\end{itemize}
```

This will start a bulleted list, but it will only reveal the first item. The second item will be revealed at the next click. The first one remains. (By the way, enumeration works exactly the same way as itemization.)

There are variations on this technique that give you more flexibility. The `\item` command has an optional argument of the form `\item<n->`. This tells LaTeX that you want that item to appear on the n^{th} overlay, and remain on the slide for all future overlays. So to reproduce the previous example, we could have coded

```
\begin{itemize}
\item<1->  My first item \pause
\item<2->  My second item
\end{itemize}
```

WARNING: You have to calculate the overlay count yourself. As far as I know, there is no auto-generated tag to solve this problem.

Using `\item<n>` instead will show that item on the n^{th} overlay and *only* on the n^{th} overlay. For example,

```
\begin{itemize}
\item<1>   My first item \pause
\item<2->  My second item \pause
\item<3->  One last item
\end{itemize}
```

produces a slide with the first item, then the second item appears and the first disappears, and finally the third appears while the second remains. Got it?

The "type=1" Option
This is handy, but a little confusing. You take this option by coding `\begin{itemize}[type=1]` and likewise for enumeration. This option puts all the "active" items on an overlay in the usual font, and sets the rest of the items in a lighter "inactive" color. If you want the old item to go *back* to the inactive faded color once you move on, you have to control this with the `\item<n>` argument.

Let's look at some examples. Consider the following code:

```
Here is my list: \pause
\begin{itemize}[type=1]
\item  First thing \pause
\item  Second thing \pause
\item  Last thing
\end{itemize}
```

What does this do? This sets the phrase "Here is my list:" and the itemized list appears with all three items in the *inactive* color. Upon the next click, "First thing" appears in the *active* color, while the remaining items are unchanged. At the next click, "Second thing" lights up, while "First thing" remains lit. Finally, "Last thing" appears in the ordinary color while the first two items remain active.

That's fine, but sometimes you will want "First thing" to go *back* to the inactive color when "Second thing" makes its appearance. As mentioned, this is controlled by specifying to which overlay the item belongs. For example, let's look at

```
Here is my list: \pause
\begin{itemize}[type=1]
\item<2>  First thing \pause
\item<3>  Second thing \pause
\item<4>  Last thing
\end{itemize}
```

This will start out like the previous example: on the second click, "First thing" will appear with all other items inactive. But this time, when "Second thing" becomes active, "First thing" will revert back to the inactive color. Finally, "Last thing" will become active and "Second thing" will go inactive.

Why is this different? Think about it: `type=1` says that the active items on an overlay are lit, while the inactive ones are dimmed. But "Second thing" is on its *own* overlay (namely, the 3$^{\text{rd}}$ one). Therefore, it is the only item to be active, while all others remain dimmed.

Even Fancier Tricks

As you become more adept at using Powerdot, you may find yourself needing to do things not described here. This is very likely, since Powerdot is very versatile and highly adaptable. For example, you will probably want to learn about the `\onslide` command. The template I have supplied will get you pretty far, but you should keep a copy of the Powerdot manual on hand. A pdf copy of the manual should come along for the ride when you download the package. If you cannot find it on your computer, you can always download a copy at `www.ctan.org`. Good luck.