\TeX provides 256 \langle dimensions \rangle and 256 \langle skips \rangle. In \texttt{CONTEXT} this is no problem, but in packages that have many authors, one can be quite sure that a lot of \langle dimensions \rangle are allocated. Packages that use \texttt{PCTeX} can therefore run out of \langle dimensions \rangle quite fast. This module was written as a reaction to persistent problems with loading \texttt{PPCHTeX} in \TeX and \texttt{PCTeX} deserves a solution. I therefore dedicate this module to Tobias Burnus and Dirk Kuypers, who use \texttt{PPCHTeX} in a \LaTeX environment and suggested a lot of extensions to the repertoire of \texttt{PPCHTeX} commands.

This module presents a solution that is quite effective: all \langle dimensions \rangle are drawn from the pool of \langle dimensions \rangle and \langle skips \rangle, depending on the availability. This is possible because \langle dimensions \rangle are \langle skips \rangle without a glue component. Therefore we can use \langle skips \rangle as \langle dimensions \rangle. However, some incompatibility can result from assignments that look like:

\begin{verbatim}
somedimen=\someskip
\end{verbatim}

In such cases the \langle dimension \rangle equals the fixed part of the \langle skip \rangle or in other words: this assignment strips off the glue. Because \texttt{PCTeX} uses no glue components, I thought I could interchange both register types without problems, but alas, this didn’t hold for all \langle dimensions \rangle.

In \texttt{PLAIN} \TeX the allocation macros are defined with \texttt{(as) \textbackslash outer}. This means that they cannot appear inside macros, not even in an indirect way. We therefore have to redefine both \texttt{\newdimen} and \texttt{\newskip} to non-\texttt{\textbackslash outer} alternatives. In most macro packages this redefinition already took place. We save the original meanings, so we can restores them afterwards.

\begin{verbatim}
1 \let\normalnewdimen = \newdimen
\let\normalnewskip = \newskip
2 \catcode'@=11 % I'd rather used \unprotect
\def\temporarynewdimen {\alloc@1\dimen\dimendef\insc@unt}
\def\temporarynewskip {\alloc@2\skip \skipdef \insc@unt}
\catcode'@=12 % and \protect.
\end{verbatim}

Here comes the trick. Depending on how many \langle dimensions \rangle and \langle skips \rangle are allocated, the \texttt{\newdimen} assigns a \langle dimensions \rangle or \langle skip \rangle. \texttt{PLAIN} \TeX allocates 15 \langle dimensions \rangle and 17 \langle skips \rangle. After loading \texttt{PCTeX}, 71 \langle dimensions \rangle and and 71 \langle skips \rangle are allocated. Indeed, \texttt{PCTeX} needs 110 \langle dimensions \rangle!

\begin{verbatim}
\def\newdimen%
  {\ifnum\count11>\count12
    \let\next=\temporarynewskip
  \else
    \let\next=\temporarynewdimen
  \fi
  \next}
\end{verbatim}

When I was testing a new version of \texttt{PPCHTeX} in \texttt{PLAIN} \TeX I had to find out that this exchange of registers sometimes leads to unwanted results. It took me some hours to find out that the source of errors originated in constructions like:

\begin{verbatim}
\ifdim\DimenOne<\DimenTwo whatever you want \else or not \fi
\end{verbatim}

When \texttt{\DimenOne} is a \langle skip \rangle and \texttt{\DimenTwo} is a \langle dimension \rangle, \TeX scans for some optional glue component, like in:

\begin{verbatim}
\skip0=\dimen0 plus 10pt minus 5pt
\end{verbatim}

The most robust solution to this problem is:

\begin{verbatim}
\ifdim\DimenOne<\DimenTwo\relax right \else wrong \fi
\end{verbatim}
PICTEX Loading Macros

Some close reading of the PICTEX source however learned me that this problem could be solved best by just honoring the allocation of \textit{dimensions} when the name of the macro explicitly stated the character sequence \texttt{dimen}. A next implementation therefore automatically declared all \textit{dimensions} with this sequence in their names with \texttt{dimen}. Again I was too optimistic, so now we do it this way (the comments are from PICTEX, which like \texttt{TABLE}, is an example of a well documented package):

\begin{verbatim}
\catcode'!='11
\temporarynewdimen\!dimenA % AWX.DVEUL..YQRST
\temporarynewdimen\!dimenB %...X.DVEU...O.QRS.
\temporarynewdimen\!dimenC %..W.X.DVEU......RS.
\temporarynewdimen\!dimenD %..W.X.DVEU......Y.RS.
\temporarynewdimen\!dimenE %..W.......G..YQ.S.
\temporarynewdimen\!dimenF %...........G..YQ.S.
\temporarynewdimen\!dimenG %...........G..YQ.S.
\temporarynewdimen\!dimenH %...........G..Y..S.
\temporarynewdimen\!dimenI %...BX.........Y....
\temporarynewdimen\!dxpos %..W......U..P....S.
\temporarynewdimen\!dypos %..WB.....U..P......
\temporarynewdimen\!xloc %..WB.....U.......S.
\temporarynewdimen\!xpos %..........L.P..Q.ST
\temporarynewdimen\!yloc %..WB.....U.......S.
\temporarynewdimen\!ypos %..........L.P..Q.ST
\temporarynewdimen\!zpt %AWBX.DVEULGP.YQ.ST
\end{verbatim}

Tobias tested this module in all kind of \LaTeX{} dialects so we were able to find out that we also needed to declare:

\begin{verbatim}
\catcode'!='12
after all, the new definition of \texttt{newdimen} became:

\begin{verbatim}
\def\newdimen#1 %
{\ifx#1\undefined
 \ifnum\count11>\count12\relax
 \temporarynewskip#1\relax
 \else
 \temporarynewdimen#1\relax
 \fi
 \edef\ascii{\meaning#1}%
 \immediate\write20{\string#1 becomes \ascii}%
 \else
 \edef\ascii{\meaning#1}%
 \immediate\write20{\string#1 already is \ascii}%
 \fi}
\end{verbatim}

Curious readers can still find the previous solution in the source. The next macro is used instead of \texttt{input}. This macro also reports some statistics.

\begin{verbatim}
\def\dimeninput#1 %
{\message{[before: d=\the\count11,s=\the\count12]}%
 \input #1 \relax
 \message{[after: d=\the\count11,s=\the\count12]}%}
\end{verbatim}

Not every package defines \texttt{fiverm}, PICTEX’s pixel, so let’s take care of that omission now:
The actual loading of \textsc{pictex} depends on the package. For \textsc{tex} users we take care of loading the auxiliary ones too.

Finally we restore the old definitions of \texttt{newdimen} and \texttt{newskip}:

\begininput
\let\newdimen = \normalnewdimen
\let\newskip = \normalnewskip

and just hope for the best.
\endinput