Cavendish-HEP 88/11

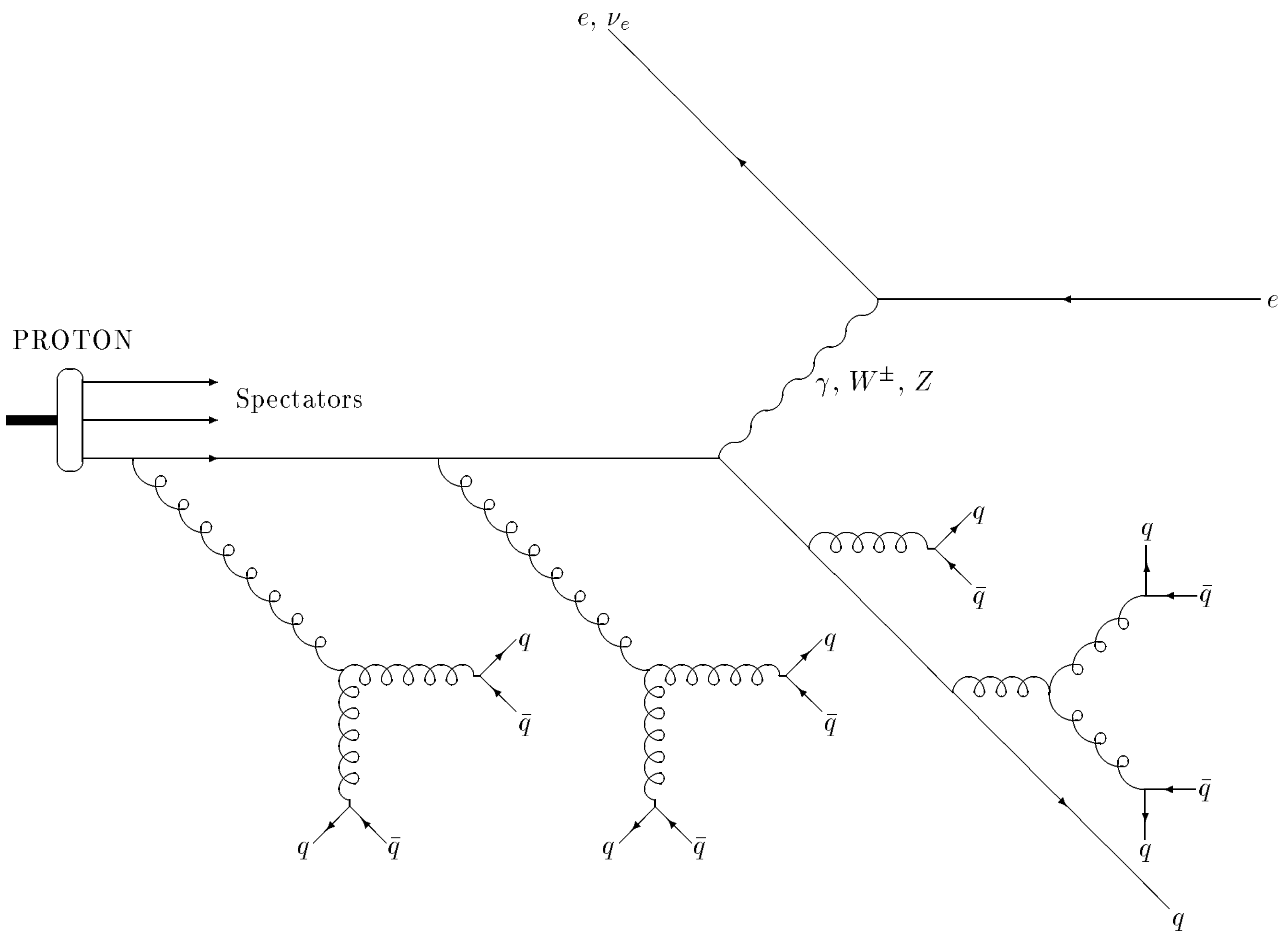
FEYNMAN

A LaTeX Routine for Generating Feynman Diagrams

Version 1.0

Michael Levine

Oct. 21, 1988



Dedication

This Routine is dedicated to the memory of
Professor Richard P. Feynman
who died on February 15, 1988.

Contents

Chapter 1

Introduction

1.1 What is FEYNMAN?

FEYNMAN is a LaTeX macropackage which allows the user to easily draw Feynman diagrams as part of the text of a LaTeX document.

1.2 What is LaTeX?

LaTeX is a TeX macropackage, that is a collection of definitions and subroutines, which modifies and expands upon the basic TeX commands by establishing a series of *environments*. In your input document these environments have the form

```
\begin{environment name}
          ⋮
\end{environment name}
```

The above was produced in the *center* environment.

FEYNMAN operates in the *picture* environment. For full details about this environment, or LaTeX in general, the LaTeX User's Guide and Reference Manual by Leslie Lamport is most highly recommended. A basic familiarity with LaTeX will be assumed in this manual.

1.3 How is FEYNMAN used?

Prior to entering the picture environment to draw a Feynman diagram the user must[1] issue the statement

```
\input FEYNMAN
```

This may be included at the beginning of the document (following \begin{document}) or immediately preceeding the first usage of **FEYNMAN** as in

```
\input FEYNMAN
\begin{picture}
      ⋮
\end{picture}
```

Of course normal picture environment commands may be used in conjunction with those of **FEYNMAN**.

[1]This is true at the time which this document is being prepared for the Cavendish Laboratories HEP Vax system, Cambridge, England.

1.4 How does FEYNMAN work?

Basically **FEYNMAN** pieces together mathematical symbols and special characters in order to draw the lines and vertices of Feynman diagrams. If you are not familiar with what a Feynman diagram is you probably do not need to use this package in any event. LaTeX has defined a number of new character fonts. The ones which are used in this macropackage draw pieces of lines and circles. These are then pieced together in certain modular configurations and stored with the *savebox* facility of LaTeX. These pieces are in turn used to construct more complicated structures and so forth. Only the basic LaTeX fonts are required, not the extended sets.

This latter point was instituted to permit greater compatibility with other systems. The disadvantage is that the basic set has an extremely limited range of characters to play with and thus there is little scope for adjusting the size and shape of particle lines which one might wish to be be drawn. For instance there are only ten different sized circles and the minimum length of a diagonal line is about half a centimeter. A further deficiency of standard LaTeX (or TeX for that matter) implementations is the small amount of internal working space which the program permits. This restricts the volume of text which may be processed at a time. Since **FEYNMAN** eats up a lot of internal memory, and a great deal more is required to store the positions of all of the individual symbols which go into drawing a complicated diagram, one is restricted in the maximum size and complexity of picture which may be drawn. This, and a possible remedy, will be further discussed in the ERRORS section at the end.

1.5 The Basics

This manual is divided into essentially three sections. The first (chapters two and three) illustrates the basic commands for drawing particle lines and vertices. The second (chapter four) details some advanced features and uses. The third consists of various appendices and the solutions to problems posed at random throughout.

In the *picture mode* of LaTeX, in which **FEYNMAN** operates, a *grid* is established with Cartesian (x, y) co-ordinates on which the picture is drawn. The scale of the grid is normally user-defined but **FEYNMAN** utilizes a default grid size in which all distances are measured in **centipoints**. One centipoint is 1/7227 of an inch or 0.14 mils or 0.0035 millimeters or 3.5 microns. This is an exceedingly small measure but there is a reason for this. All numbers input to **FEYNMAN**'s parameters, in particular lengths and positions, must be integers. You will probably find it most convenient to think of lengths in terms of thousands of centipoints; a thousand centipoints equalling one seventh of an inch or one third of a centimeter. The eye is quite capable of detecting displacements of one centipoint (or even less) even if the system printer is incapable of such resolution. Printers will, however, round positions to the nearest pixel and accurate positioning will aid correct rounding.

When drawing a Feynman diagram the user should have a rough sketch of the picture, including labelling, arrows *etc.* He or she should then select a *box size*, in centipoints, into which the picture should be placed. This will determine size of the block allocated on your page, possibly between text, equations or other pictures, which will be given to the current picture. One then issues the command:

```
\begin{picture}(7000,7000)
```

which enters picture mode and sets the box size to be 7000×7000, or roughly 1 inch by 1 inch. Any box size may be used, up to the declared size of the page. When **FEYNMAN** is input it automatically sets the scale and this will be retained unless re-defined by the user. *This cannot be done while* **FEYNMAN** *is in effect or disaster will ensue.*

Next the user must select a co-ordinate, generally (0,0), from which to start drawing the diagram. You need not keep track of the numerical co-ordinates thereafter, **FEYNMAN** will

do that for you. In fact that is the whole point of the program. You tell it what you want drawn, in which direction and how long (in some sense) it should be and it will do it. After drawing a line (or vertex) all of the important positions will be given names and you can specify *these* to tell it where to commence the next line. In point of fact you can usually begin your diagram at any point in it and fill the lines in many different orders. A little experience will show that it is usually easiest to start at one end, or branch, and work your way along. This way the same position names may be re-used, superseding the old values. The details will be given in chapter 2. When the picture is complete the command:
\end{picture}
is issued.

The basic command for drawing lines is \drawline. With this all standard line and vertex types may be drawn. In addition a number of specialized vertices have been pre-defined. These are drawn with the \drawvertex command which is discussed in chapter three.

1.6 Getting Started

A basic familiarity with **FEYNMAN** may be obtained by reading chapter one and sections 2.1-2.7 and 2.9. The following small sample file illustrates the basic form of a file utilizing **FEYNMAN**.

```
\documentstyle[12pt]{article}
\begin{document}
\input FEYNMAN
\textheight 800pt \textwidth 450pt
The gluon then branches into a pair of partons, in this case a quark-antiquark
pair:

\begin{picture}(10000,18000)
\drawline\gluon[\S\REG](0,16000)[8]
\put(2500,\pmidy){Times 3 colours}
\drawline\fermion[\SW\REG](\gluonbackx,\gluonbacky)[2000]
\drawline\fermion[\SE\REG](\gluonbackx,\gluonbacky)[2000]
\end{picture}
\hskip 1.38 in
\begin{picture}(10000,18000)
\drawline\gluon[\S\CENTRALGLUON](0,16000)[8]
\drawline\scalar[\SW\REG](\gluonbackx,\gluonbacky)[5]
\drawline\scalar[\SE\REG](\gluonbackx,\gluonbacky)[5]
\end{picture}

Where the second picture is the interaction with the hypothesized squark.
\end{document}
```

A few brief points should be made about the above. The *textheight* and *textwidth* set the pagesize, made larger than usual in the above in order to draw longer and wider pictures. The *put* command is a basic LaTeX command for placing text in a picture. The commands *hskip,* *vskip* may be used for adding extra space, horizontally and vertically, where needed. Any size ([10pt],[11pt] or [12pt]) and style of document may be specified and the resultant diagrams will be unaffected with one exception.[2] The reader might wish to duplicate the above example and run it to see the results.

Running and printing a LaTeX file is a system-dependent operation. On the Cavendish Labs Vax, where this program was developed, the syntax is: LATEX <filename>
with default filetype 'TEX'. To print out the compiled file on the LN03 laser printer the commands:
DVI2LN3 <filename>.DVI /H=337.5 /V=200
PRINT/QUEUE=LN03 <filename>.LN3

would be issued. It is generally easier to define a command file, called PRINTLATEX.COM, or some such, of the form:

```
$! To print out a Latex file
$ LATEX 'P1'
$ DVI2LN3 'P1'.DVI /H=337.5 /V=200  ! /s=14 /n=1
$ PRINT/QUEUE=LN03 'P1'.LN3
$ DELETE 'P1'.DVI;*
$ DELETE 'P1'.LIS;*
$ DELETE 'P1'.AUX;*
```

which would be executed by @PRINTLATEX <filename> and would further delete the additional files which LaTeX will place on your disk. On other computer systems the implementation will be somewhat different.

The syntax of the various **FEYNMAN** commands will now be presented.

[2]The exception is that when horizontal photons are drawn in a [12pt] document, that the command "\bigphotons" must appear somewhere between the \input FEYNMAN statement and the first time that such a photon is drawn.

Chapter 2

Drawing Lines

2.1 The \drawline Command

The principal command for drawing a particle line in **FEYNMAN** is via the *drawline* statement. The syntax is:

```
\drawline<particle type>[<particle direction><configuration>]
         <(x,y), the co-ordinates of the beginning of the line>[<length>]
```

where it is understood that the triangular brackets do not appear in the syntax but their contents describe what goes in that position. An example would be:

```
\drawline\photon[\NW\FLIPPED](1500,-18000)[6]
```

which would draw a photon (or W,Z) starting from the point (1500,-18000), with distances in centipoints (section 1.5), in the NorthWesterly direction (*i.e.* towards the upper lefthand corner), in a *flipped* configuration (described below) for an extent of 6 'units'. In the case of photons each unit is a 'half-wiggle'. The actual distance traversed on the page is not specified since photons come in distinct 'quanta', in this case half-wiggles.

When the above statement is issued a number of parameters are returned. These may be used by the artiste (this is 'Feyn Art'). These are:

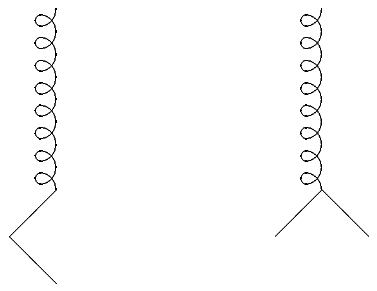
```
\photonfrontx,\photonfronty:       The (x,y) co-ordinates of the front of the line.
\photonbackx,\photonbacky:         The (x,y) co-ordinates of the back of the line.
\photonlengthx,\photonlengthy:     The (x,y) extent of the line.
\photoncount                       The number of photons printed thus far.
\particlefrontx,\particlefronty:   The (x,y) co-ords of the front of the line.
\particlemidx,\particlemidy:       The (x,y) co-ordinates of the middle of the line.
\particlebackx,\particlebacky:     The (x,y) co-ordinates of the back of the line.
\particlelengthx,\particlelengthy: The (x,y) extent of the line.
```

All of the terms beginning with '\particle' may be abbreviated to '\p' plus the rest of the name. For instance, \pmidx is the same as \particlemidx *etc.* You may be wondering why there is some duplication, such as \pfrontx and \photonfrontx. The reason is that terms beginning with *particle* refer to the last particle line drawn by \drawline whereas those commencing with *photon* refer only to the last *photon line* drawn. This distinction often proves to be most convenient.

To illustrate this point consider the following combination of statements:

```
\input FEYNMAN

\vskip -0.7in
\begin{picture}(18000,8000)
\drawline\gluon[\S\CURLY](0,0)[8]
\drawline\fermion[\SW\REG](\particlebackx,\particlebacky)[2500]
\drawline\fermion[\SE\REG](\particlebackx,\particlebacky)[2500]
\drawline\gluon[\S\CURLY](10000,0)[8]
\drawline\fermion[\SW\REG](\particlebackx,\particlebacky)[2500]
\drawline\fermion[\SE\REG](\gluonbackx,\gluonbacky)[2500]
\end{picture}

\vskip 1.5in
```

which produces the following results:



We see immediately what the difference is. In the first diagram \particlebackx,y refers, first to the gluon after it is drawn, and then to the \SW fermion after that has been drawn. If a third reference to those co-ordinates had been made it would have referred to the final co-ordinates of the \SE fermion. The second gluon line begins 10000 centipoints to the right of the first. This time the \SE fermion commences at \gluonbackx,y which is the end co-ordinates of the gluon and will remain so until the end of the picture of until another gluon is drawn.

We now proceed to detail the various parameters of the \drawline command given above.

2.2 The Types of Particle Lines

The four basic kinds of particle lines which **FEYNMAN** can draw are given by the following input parameters:

```
\fermion
\scalar
\photon
\gluon
```

A fifth category, `\especial`, which allows the user to define his own non-standard style of line, has not yet been implemented. As we have seen this is the first input parameter of \drawline and is implemented as

```
\drawline\fermion...
\drawline\scalar...
\drawline\photon...
\drawline\gluon...
```

Note that this is the only argument of \drawline which is in lowercase letters. The specifics of each of these will be given in separate sections.

2.3 Particle Direction

Each particle line may be drawn in any of eight possible directions, specified by the points of the compass with North always understood as being at the top of the page:

```
\N   \NE   \E   \SE   \S   \SW   \W   \NW
```

This is the second argument of \drawline.

```
\drawline\photon[\NW...
```

and so forth. The line is drawn from the specified *front* point in the indicated direction. Note that all directions are in uppercase and don't forget the backslash!

2.4 Line Configuration

Many different particle types are available in a variety of forms. For the specific forms available see the section on the appropriate particle type below. The current configurations available are:

```
\REG
\FLIPPED
\CURLY
\FLIPPEDCURLY
\FLAT
\FLIPPEDFLAT
\CENTRAL
\FLIPPEDCENTRAL
\LONGPHOTON
\FLIPPEDLONG
\SQUASHEDGLUON
```

The shape of the individual particle types will be presented shortly but a few general words might be given here. First, not every particle type exists in all configurations in all directions. \REG is something of a default (although `\REG` must still be included as a parameter in \drawline) and is defined for all particles in all directions. It may not be your own æsthetic favourite, however. \FLIPPED lines are inverted about their axes with respect to the \REG lines, but are otherwise the same. Obviously, since fermions and scalars are merely straight lines, the \REG and \FLIPPED configurations will be the same. In fact for these two particle types \REG is the **only** legitimate configuration (but see \THICKLINES at the conclusion of this section). The other configurations do not, in general, exist for all of the directions available to a particle. \FLIPPEDCURLY, \FLIPPEDCENTRAL, \FLIPPEDLONG and \FLIPPEDFLAT are flipped along the direction axis when compared with \CURLY, \CENTRAL, \LONGPHOTON and \FLAT respectively. \CENTRAL and \FLIPPEDCENTRAL exist only for gluons (\gluon), \LONGPHOTON and \FLIPPEDLONG exist only for photons, while \SQUASHED exists only for gluons in the \E direction!

The line configuration is the third input to \drawline and follows the direction within the square brackets with no intervening spaces:

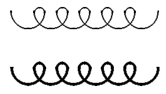
```
\drawline\scalar[\S\REG]...
\drawline\gluon[\NW\FLIPPED]...
```

and so forth.

A final word concerns the \THINLINES and \THICKLINES commands. When issued prior to a \drawline or \drawvertex statement they cause the particle lines to be drawn in a thin or thick fashion This remains in effect until superseded or the end of the picture is encountered. The default is \THINLINES. These commands are identical to the LaTeX \thinlines and \thicklines commands except when drawing photons in the `\E` and `\W` directions (any configuration). These will not be emboldened by \thicklines but will be by \THICKLINES. Photons in the `\N` and `\S` directions (any configuration) are unaffected since they are drawn in an intermediate state. The following illustrates the difference:

```
\begin{picture}(8000,2000)
\drawline\gluon[\E\FLIPPEDCENTRAL](0,0)[5]
\end{picture}

\begin{picture}(8000,2000)
\THICKLINES
\drawline\gluon[\E\FLIPPEDCENTRAL](0,0)[5]
\end{picture}
```

which produces



2.5 Line Co-ordinate Parameters

2.5.1 Input Parameters

The fourth and fifth arguments of the \drawline command are the (x,y) co-ordinates of the *beginning* of the particle line. these are as measured in *centipoints* on the grid which **FEYNMAN** has established. They are entered in the format (*x co-ordinate*, *y co-ordinate*) where x and y may be integer numbers (between, roughly, -30,000 and +30,000) or variables (counters) with numerical values. A number of variables have been pre-defined and available for use. The user may also define his own (see section 4.2.1 on storing information). Some samples may be illustrative:

```
\drawline\fermion[\SE\REG](-1500,12000)[2000]
\drawline\gluon[\E\SQUASHEDGLUON](\photonbackx,\photonbacky)[2]
\drawline\scalar[\N\REG](\Xone,\Yone)[11]
\drawline\photon[\S\FLIPPEDFLAT](3000,\pmidy)[7]
```

In the above \photonbackx and \photonbacky are co-ordinates, presumably returned from a previously drawing a photon. \Xone and \Yone are some values stored by the user and \pmidy is the ordinate of the midpoint of the previously drawn line. Note in the last example how the x co-ordinate is a number while the y co-ordinate is a variable.

2.5.2 Output Parameters

As discussed in section 2.1 a number of useful positional parameters are returned when \drawline is called. For the appropriate lines the following are defined, each definition superseding the previous value of the variable.

For all lines:

```
\particlefrontx,\particlefronty: The (x,y) co-ords of the front of the line.
\particlemidx,\particlemidy:     The (x,y) co-ordinates of the middle of the line.
\particlebackx,\particlebacky:   The (x,y) co-ordinates of the back of the line.
```

For photons:

```
\photonfrontx,\photonfronty:     The (x,y) co-ordinates of the front of the photon.
\photonbackx,\photonbacky:       The (x,y) co-ordinates of the back of the photon.
```

For gluons:

```
\gluonfrontx,\gluonfronty:       The (x,y) co-ordinates of the front of the gluon.
\gluonbackx,\gluonbacky:         The (x,y) co-ordinates of the back of the gluon.
```

For fermions:

```
\fermionfrontx,\fermionfronty:   The (x,y) co-ordinates of the front of the fermion.
\fermionbackx,\fermionbacky:     The (x,y) co-ordinates of the back of the fermion.
```

For scalars:

```
\scalarfrontx,\scalarfronty:     The (x,y) co-ordinates of the front of the scalar.
\scalarbackx,\scalarbacky:       The (x,y) co-ordinates of the back of the scalar.
```

As an illustration the command
```
 \drawline\fermion[\E\REG](0,0)[2000]
```
has

```
     \pfrontx=0        \pfronty=0
     \pmidx=1000       \pmidy=0
     \pbackx=2000      \pbacky=0
```

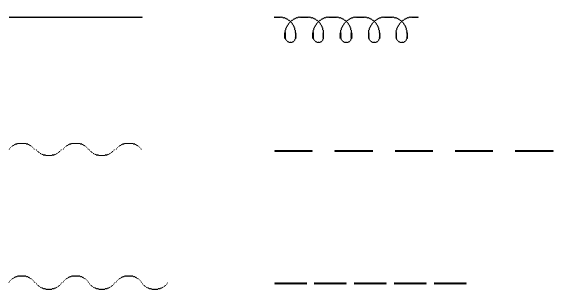
2.6 Line Length

2.6.1 Input Parameters

The final parameter to be given to \drawline is the *length* of the particle to be drawn. The units in which the length is given vary between the four particle types. For fermions it is simply the actual length in centipoints. For scalars it is the number of line segments, each of which is separated by a *gap*. The scalar line always begins and ends on a line segment, never a gap. The default lengths of both segments and gaps may be overruled by the user (see the section on scalars). For gluons the length parameter is the *number of loops*. The actual length of each loop depends upon which style is selected and whether the gluon is drawn diagonally (at a slant) or not. For photons the unit of measure is not a 'wiggle', but a 'half-wiggle'. This enables one to produce a photon which both begins and ends on the 'upward' (or 'downward') part of its oscillation. This is illustrated in the following program:

```
\begin{document}
\input FEYNMAN
\textheight 600pt \textwidth 400pt

\begin{picture}(20000,10000)
\drawline\fermion[\E\REG](0,10000)[5000]
\drawline\gluon[\E\REG](10000,10000)[5]
\drawline\photon[\E\REG](0,5000)[5]
\drawline\photon[\E\REG](0,0)[6]
\drawline\scalar[\E\REG](10000,5000)[5]
\seglength=1200   \gaplength=300   %  Changes the \scalar defaults.
\drawline\scalar[\E\REG](10000,0)[5]
\end{picture}
\end{document}
```

which produces:



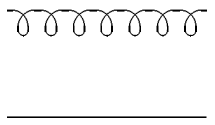
2.6.2 Output Parameters

In the same way as co-ordinates related to the particle are returned, the *displacements* of the line are also given. These are essentially the $(\Delta x, \Delta y)$ between the ending and beginning co-ordinates. Thus a particle drawn in the \N direction will have positive \particlelengthy (\plengthy for short) and zero \particlelengthx. A particle in the \SE direction will have a positive \particlelengthx but a negative \particlelengthy (and these will be of the same magnitude). The full list is:

`\particlelengthx,\particlelengthy:`	`The (x,y) extent of the line.`
`\photonlengthx,\photonlengthy:`	`The (x,y) extent of the photon.`
`\gluonlengthx,\gluonlengthy:`	`The (x,y) extent of the gluon.`
`\scalarlengthx,\scalarlengthy:`	`The (x,y) extent of the scalar.`
`\fermionlengthx,\fermionlengthy:`	`The (x,y) extent of the fermion.`
`\fermionlength`	`The total length of the fermion.`

A common usage of this parameter would be to draw a fermion line to be the same length as, say, a gluon line.

```
\begin{picture}(9000,4500)
\drawline\gluon[\E\REG](0,4000)[7]
\drawline\fermion[\E\REG](0,0)[\gluonlengthx]
\end{picture}
```



Each time that a new line is drawn \plengthx,y are re-assigned. \photonlengthx,y is only re-assigned when the next photon is drawn, \gluonlengthx,y is only re-assigned when the next gluon is drawn *etc.*

2.7 Fermions

The format for drawing a fermion with \drawline is:

```
\drawline\fermion[<fermion direction>\REG]
       <(x,y), the co-ordinates of the beginning of the fermion>[<length(cpt)>]
```

where the arguments have been described in the preceding sections. When **FEYNMAN**draws a fermion it uses the \line facility of LaTeX. This places a restriction upon diagonally drawn fermions (those in the \NE, \SE, \NW and \SW directions). The minimum length of such lines is 1416 centipoints. The user can draw fermions in other directions (NNE *etc.*) by utilizing the \put command of LaTeX. The minimum extension of these lines is at least as great. Fermions drawn in the N,S,E and W directions have no such restrictions. If attempts are made to draw a fermion of sub-minimal length it will be as though it were invisible. That is all of the spacing and parameters will be as before but a blank space will appear on the page. This effect may be deliberately achieved for *any* particle or vertex using the *phantom* facility discussed under advanced features.

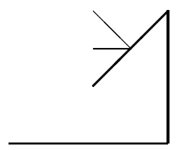
The only option really available for fermions is the use of \thicklines and \thinlines as described at the conclusion of section 2.4. The parameters returned by \drawline when a fermion is drawn are:

```
\fermionfrontx,\fermionfronty:     The (x,y) co-ordinates of the front of the line.
\fermionbackx,\fermionbacky:       The (x,y) co-ordinates of the back of the line.
\fermionlengthx,\fermionlengthy:   The (x,y) extent of the line.
\fermionlength                     The total length of the line.
\fermioncount                      The number of fermions printed thus far.
\particlefrontx,\particlefronty:   The (x,y) co-ords of the front of the line.
\particlemidx,\particlemidy:       The (x,y) co-ordinates of the middle of the line.
\particlebackx,\particlebacky:     The (x,y) co-ordinates of the back of the line.
\particlelengthx,\particlelengthy: The (x,y) extent of the line.
```

As an example:

```
\begin{picture}(8000,8000)
\drawline\fermion[\E\REG](0,0)[6000]
\THICKLINES
\drawline\fermion[\N\REG](\fermionbackx,\pbacky)[5000]
\drawline\fermion[\SW\REG](\fermionbackx,\fermionbacky)[4000]
\THINLINES
\drawline\fermion[\NW\REG](\pmidx,\pmidy)[2000]
\drawline\fermion[\W\REG](\pfrontx,\pfronty)[\fermionlengthy]
\end{picture}
```

which produces:



2.8 Scalars

2.8.1 The Anatomy of a Scalar

A scalar particle (eg. a Higgs) is generally drawn as a dashed line, that is a series of line *segments* separated by *gaps*. The format for drawing a scalar with \drawline is:

```
\drawline\scalar[<scalar direction>\REG]
       <(x,y), the co-ordinates of the beginning of the scalar>
       [<number of segments>]
```

We note the similarity between the fermion and scalar lines. The principal difference is in the units of length. Here the extent of the line is given by an integral number of segments. These will be of equal length and equally spaced. The same limitations on the length of fermion lines applies here: diagonal (NW, NE *etc.*) line segments may not be shorter than 1416 centipoints, which is the default length for all directions. The default gap length is 850 centipoints and there is no innate limitation on this value (other than it must not be negative or larger than the page size). Attempts to draw a slanted scalar with segments of under 1416 centipoints will lead to a blank space in the diagram.

The parameters returned by \drawline when a scalar is drawn are:

```
\scalarfrontx,\scalarfronty:       The (x,y) co-ordinates of the front of the scalar.
\scalarbackx,\scalarbacky:         The (x,y) co-ordinates of the back of the scalar.
\scalarlengthx,\scalarlengthy:     The (x,y) extent of the scalar.
\scalarcount:                      The total number of scalars printed thus far.
\particlefrontx,\particlefronty:   The (x,y) co-ords of the front of the line.
\particlemidx,\particlemidy:       The (x,y) co-ordinates of the middle of the line.
\particlebackx,\particlebacky:     The (x,y) co-ordinates of the back of the line.
\particlelengthx,\particlelengthy: The (x,y) extent of the line.
```

2.8.2 Adjusting the Spacing

The segment length is given, in centipoints, by the variable \seglength and the gap length by \gaplength. Both are adjustable by the user by issuing the commands:

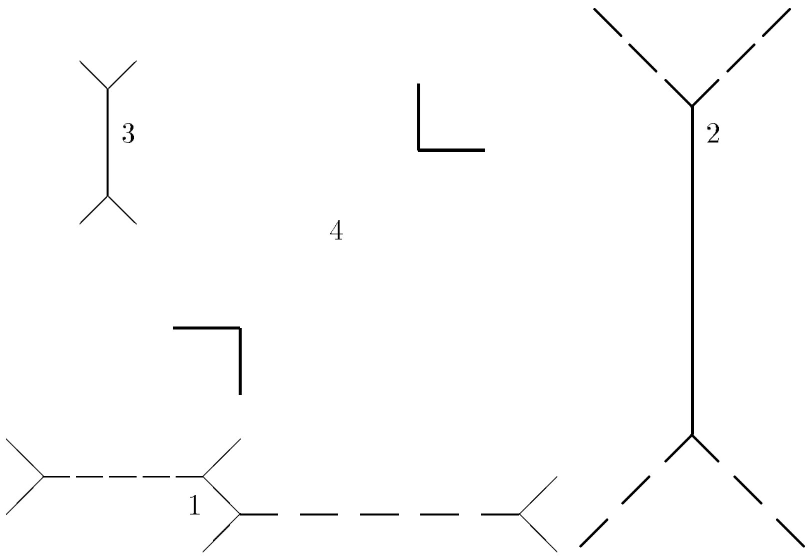
```
\global\seglength=<your length in centipoints>
\global\gaplength=<your length in centipoints>
```

just prior to a \drawline\scalar command. After this they automatically return to their default values of 1416 and 850. If only one is adjusted the other will retain its default value. The modifiers \thicklines and \thinlines work normally here with \thinlines the default (but \thicklines will remain in effect until \thinlines or \end{picture} is encountered).

These features are illustrated by the following sample file:

```
\begin{picture}(25000,25000)
% First picture:
\drawline\scalar[\E\REG](6000,0)[5]
\drawline\fermion[\SE\REG](\scalarbackx,\scalarbacky)[2000]
\drawline\fermion[\NE\REG](\scalarbackx,\scalarbacky)[2000]
\drawline\fermion[\SW\REG](\scalarfrontx,\scalarfronty)[2000]
\drawline\fermion[\NW\REG](\scalarfrontx,\scalarfronty)[2000]
\global\seglength=1000  \global\gaplength=250
\drawline\scalar[\W\REG](\pbackx,\pbacky)[5]
\drawline\fermion[\SW\REG](\particlebackx,\particlebacky)[2000]
\drawline\fermion[\NW\REG](\scalarbackx,\scalarbacky)[2000]
\drawline\fermion[\NE\REG](\scalarfrontx,\scalarfronty)[2000]
\put(4000,0){1}

% Second Picture:
\THICKLINES
\drawline\fermion[\N\REG](23000,3000)[12345]
\global\gaplength=473
\drawline\scalar[\NW\REG](\pbackx,\fermionbacky)[3]
\global\gaplength=473
\drawline\scalar[\NE\REG](\fermionbackx,\pfronty)[3]
\drawline\scalar[\SW\REG](\fermionfrontx,\fermionfronty)[3]
\drawline\scalar[\SE\REG](\fermionfrontx,\fermionfronty)[3]
\put(23500,14000){2}

% Third Picture:
\THINLINES
\global\gaplength=0   \global\seglength=1000
\drawline\scalar[\S\REG](1000,16000)[4]
\drawline\fermion[\NW\REG](1000,16000)[1500]
\drawline\fermion[\NE\REG](1000,16000)[1500]
\drawline\fermion[\SW\REG](1000,12000)[1500]
\drawline\fermion[\SE\REG](1000,12000)[1500]
\put(1500,14000){3}

% Fourth Picture:
\global\gaplength=500  \global\seglength=500
\thicklines
\drawline\scalar[\NE\REG](6000,7000)[10]
\put(\pmidx,\pmidy){4}
\drawline\fermion[\W\REG](\scalarfrontx,\scalarfronty)[2500]
\drawline\fermion[\S\REG](\scalarfrontx,\scalarfronty)[2500]
\drawline\fermion[\E\REG](\scalarbackx,\scalarbacky)[2500]
\drawline\fermion[\N\REG](\scalarbackx,\scalarbacky)[2500]
\end{picture}
```

which produces:



There are a few points of note. In the first picture observe how \pbackx,y (≡\particlebackx,y) is used as opposed to \scalarbackx,y. For more on this point see section 2.1. Also note that the \seglength assignments stay in effect until the next scalar is drawn. The second figure illustrates how \gaplength automatically reverts to its default and further demonstrates the assignments of the co-ordinate variables. The third diagram shows that a zero gaplength creates a fermion and that numerical co-ordinates may be used just as easily as the pre-assigned positioning variables. This is also illustrated by the positioning of the numerical labels (1,2,3,4) which was accomplished with minimal thought. Finally the fourth diagram shows what happens when the value of \seglength is reduced to less that 1416 for a diagonal (non-horizontal, non-vertical) scalar. It also demonstrates how to locate the central point of a line.

2.9 Photons

2.9.1 The Anatomy of a Photon

A photon line consists of a series of 'wiggles' or undulations. The length of each undulation depends upon the style of photon selected. The format for drawing a photon with \drawline is:

```
\drawline\photon[<photon direction><photon style>]
        <(x,y), the co-ordinates of the beginning of the photon>
        [<number of HALF-wiggles>]
```

The reason for specifying the number of *half*-wiggles is that it is sometimes convenient to have a line begin and end in either orientation (see section 2.6.1 for an example of this).

The styles currently available for photons are:

```
\REG
\FLIPPED
\CURLY
\FLIPPEDCURLY
\FLAT
\FLIPPEDFLAT
\LONGPHOTON
\FLIPPEDLONG
```

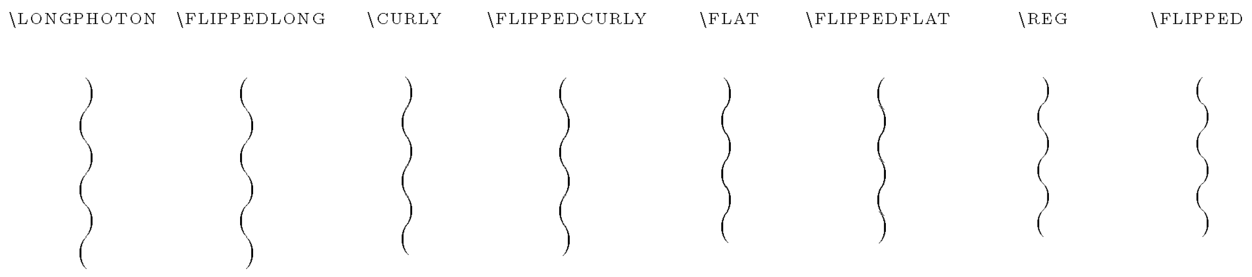
The \REG and \FLIPPED styles are available in all orientations. The \LONGPHOTON and \FLIPPEDLONG photons are only available in the \N and \S directions. The others are available only in the following directions: `\N, \S, \NE, \SE, \NW, \SW`. The \thicklines option is only available to photons drawn in the `\NE, \SW, \SE` and `\NW` directions, however \THICKLINES is also available in the `\E` and `\W` orientations and has the same effect. Attempts to draw a combination other than these will result in an error message on the screen and in the <jobname>.lis file. These options will be illustrated in the next subsection.

The parameters returned by \drawline\photon are analogous to those returned after drawing fermions and scalars. These are:

`\photonfrontx,\photonfronty:`	`The (x,y) co-ordinates of the front of the line.`
`\photonbackx,\photonbacky:`	`The (x,y) co-ordinates of the back of the line.`
`\photonlengthx,\photonlengthy:`	`The (x,y) extent of the line.`
`\photoncount`	`The number of photons printed thus far.`
`\particlefrontx,\particlefronty:`	`The (x,y) co-ords of the front of the line.`
`\particlemidx,\particlemidy:`	`The (x,y) co-ordinates of the middle of the line.`
`\particlebackx,\particlebacky:`	`The (x,y) co-ordinates of the back of the line.`
`\particlelengthx,\particlelengthy:`	`The (x,y) extent of the line.`

2.9.2 Examples and Details

The different styles of photon are:



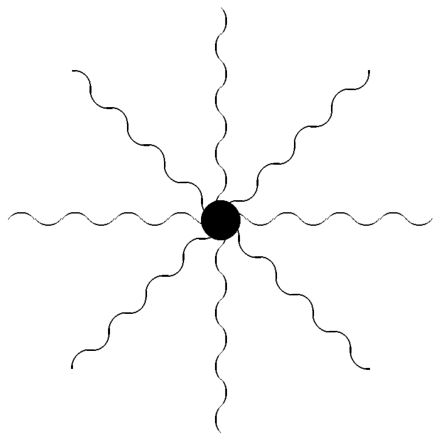
The different directions may be illustrated by the following file:

```
% EXAMPLE OF USING FEYNMAN TO DRAW PHOTON DIAGRAMS IN TEX.
\documentstyle [11pt]{article}
\input FEYNMAN
\begin {document}
```

```
\centerline{PHOTONBURST using FEYNMAN}
\vskip 0.75in
\hskip 1.2in
\begin{picture}(20000,20000)(-10000,-10000)
\put(0,0){\circle*{1500}}
\drawline\photon[\N\REG](0,0)[8]
\drawline\photon[\NE\CURLY](0,0)[8]
\drawline\photon[\E\REG](0,0)[8]
\drawline\photon[\SE\CURLY](0,0)[8]
\drawline\photon[\S\REG](0,0)[8]
\drawline\photon[\SW\CURLY](0,0)[8]
\drawline\photon[\W\REG](0,0)[8]
\drawline\photon[\NW\CURLY](0,0)[8]
\end{picture}
\end{document}
```

Which produces:

PHOTONBURST using FEYNMAN

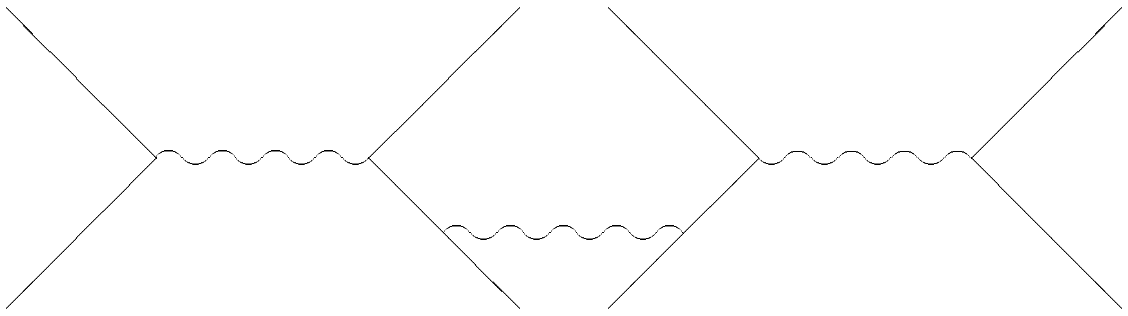


A number of points may be noted. The first is the additional argument in the \begin{picture} statement. This sets the lower left-hand corner of the picture box to the co-ordinates (-10000,-10000) in centipoints. Thus far this has been omitted and this corner is assigned the default co-ordinate of (0,0). The next point is the use of \circle*. This draws a disk centred at the specified spot of the demanded diameter. Only small disks can be thus drawn. Also note that all of the photons begin their curvature in a *clockwise* sense. Similarly the \flipped versions begin in a counter-clockwise orientation. In passing, the \centerline command may be used as an alternative to the LaTeX centered environment.

The next example illustrates a case where it becomes useful to be able to draw an odd number of half-wiggles.

```
\documentstyle [12pt]{article}
\input FEYNMAN
\begin {document}
\bigphotons
\begin{picture}(10000,10000)(0,0)
\drawline\photon[\E\REG](0,0)[8]  % Even number of half-wiggles.
\drawline\fermion[\NW\REG](\photonfrontx,\photonfronty)[\photonlengthx]
    % Make the fermions the same length as the photon.
\drawline\fermion[\SW\REG](\photonfrontx,\photonfronty)[\photonlengthx]
\drawline\fermion[\NE\REG](\photonbackx,\photonbacky)[\photonlengthx]
\drawline\fermion[\SE\REG](\photonbackx,\photonbacky)[\photonlengthx]
\global\divide\fermionlength by 2  %  Halves \fermionlength
\drawline\photon[\E\REG](\pmidx,\pmidy)[9]  % Odd number of half-wiggles.
\drawline\fermion[\SW\REG](\photonbackx,\photonbacky)[\fermionlength]
    % Draws fermion at the halved value of \fermionlength...which is
    % Half of the value of the previous fermions.
\drawline\fermion[\NE\REG](\photonbackx,\photonbacky)[\fermionlength]
\drawline\photon[\E\FLIPPED](\pbackx,\pbacky)[8]  % Even number of half-wiggles.
\drawline\fermion[\NW\REG](\photonfrontx,\photonfronty)[\photonlengthx]
    % Make the fermions the same length as the photon.
\drawline\fermion[\NE\REG](\photonbackx,\photonbacky)[\photonlengthx]
\drawline\fermion[\SE\REG](\photonbackx,\photonbacky)[\photonlengthx]
\end{picture}
\vskip 1in
\end{document}
```

Which draws:



Let us analyse this. The first new feature that we see is the "\bigphotons" statement. This statement is only required when photons are going to be drawn either in the \E or \W directions **and** the document size has been selected to be [12pt]. It is best to include it automatically if any Feynman diagrams are to be drawn in a 12pt document. It should appear somewhere after the \input FEYNMAN statement and before the first `\drawline\photon[\E...` or `\drawline\photon[\W...` command. This is the only instance when it is used.

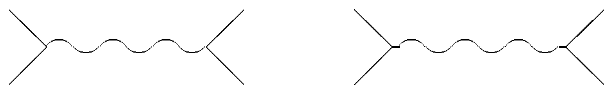
The next thing we observe is how the \photonlengthx was used to draw the fermions and photons to the same length. This technique could not be used to draw, say, gluons and photons to the same length. Because of the angle between the second (middle) photon line and the fermion legs to which it is attached it was desirable to draw both ends of the photon as 'down-turning'. If the connecting photon had been drawn between the upper fermion lines then, instead of the middle photon being

`\drawline\photon[\E\REG](\pmidx,\pmidy)[7]` it would have been `\drawline\photon[\E\FLIPPED](\pmidx,\pmidy)[7]`.

The next new item is the "\global\divide" command. This will be further discussed in chapter four but its function here is obvious. We wish to draw a fermion line centered at the end of the middle photon's right (East) end. The easiest way to do this is to draw fermions of half of the desired length in opposite directions. To obtain this value we use the statement `\global\divide\fermionlength by 2` which reduces the value of this variable by a factor of two. The divisor must be integral and the quotient will be rounded to an integer. See the section on information storage for how to record the value of \fermionlength prior to halving it. This technique could just as easily been used if, instead of a fermion, we had had a photon or gluon (with an even number of loops or half-wiggles). In these case one of the two halves would have been *flipped* with respect to the other. Can you see why a scalar could not be drawn this way?

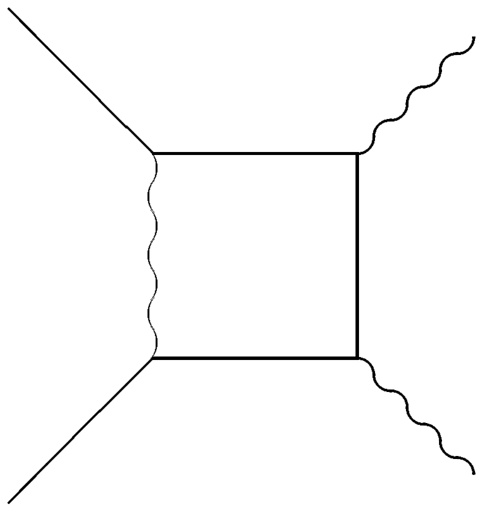
The final point is that the third (right-most) photon was drawn in a \flipped configuration in order to give a left-right symmetry to the diagram. The user is encouraged to try the following exercises. Firstly draw the above but with an even number of wiggles in all three photons to see the difference. Secondly try to draw the picture where photons connect both the upper and lower fermion pairs, creating a loop. Finally try to rotate the diagram through 45°.

Aside from \bigphotons discussed above there is one other photonic feature which we will mention in passing. Photons may be drawn with *stems* on them. This is an advanced feature which will be discussed in chapter four and an example illustrating the difference between a stemmed and unstemmed line will suffice for the present:



UNSTEMMED STEMMED

Exercise: The following diagram has eight lines. Using **FEYNMAN** duplicate it using only eight commands.



2.10 Gluons

2.10.1 Anatomy of a Gluon

A gluon consists of connected loops. As in the case of the photon line, the length of a single loop varies between styles and configurations. The syntax for creating a gluon with the \drawline command is:

```
\drawline\gluon[<gluon direction><gluon style>]
       <(x,y), the co-ordinates of the beginning of the gluon>[<number of loops>]
```

The styles currently available for gluons are:

```
\REG
\FLIPPED
\CURLY
\FLIPPEDCURLY
\FLAT
\FLIPPEDFLAT
\CENTRAL
\FLIPPEDCENTRAL
\SQUASHEDGLUON
```

The \REG and \FLIPPED styles are available in all orientations. The others are available only in the following directions:

\CURLY:	\N, \S, \E, \W
\FLIPPEDCURLY:	\N, \S, \E, \W
\CENTRAL:	\N, \S, \E, \W
\FLIPPEDCENTRAL:	\N, \S, \E, \W
\FLAT:	\E, \W
\FLIPPEDFLAT:	\E, \W
\SQUASHEDGLUON:	\E

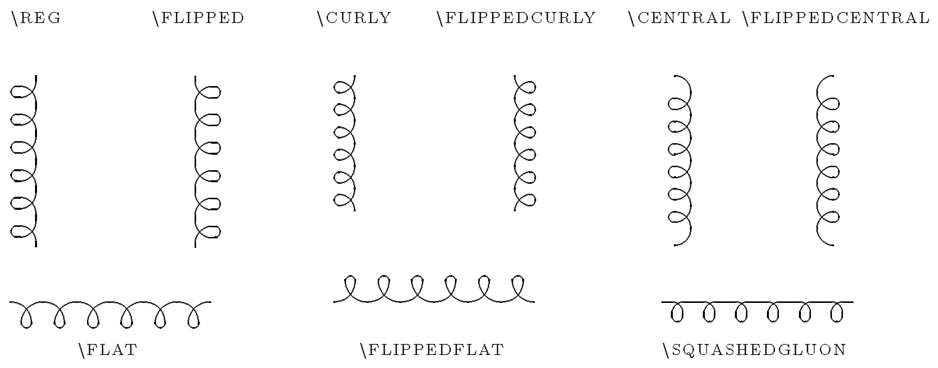
The differences in style are illustrated in the next subsection. The \THICKLINES, \THINLINES, \thicklines and \thinlines options are available in all orientations and styles.

The parameters returned by \drawline\gluon are analogous to those returned after drawing photons, fermions and scalars:

\gluonfrontx,\gluonfronty:	The (x,y) co-ordinates of the front of the line.
\gluonbackx,\gluonbacky:	The (x,y) co-ordinates of the back of the line.
\gluonlengthx,\gluonlengthy:	The (x,y) extent of the line.
\gluoncount	The number of gluons printed thus far.
\particlefrontx,\particlefronty:	The (x,y) co-ords of the front of the line.
\particlemidx,\particlemidy:	The (x,y) co-ordinates of the middle of the line.
\particlebackx,\particlebacky:	The (x,y) co-ordinates of the back of the line.
\particlelengthx,\particlelengthy:	The (x,y) extent of the line.

2.10.2 Examples and Details

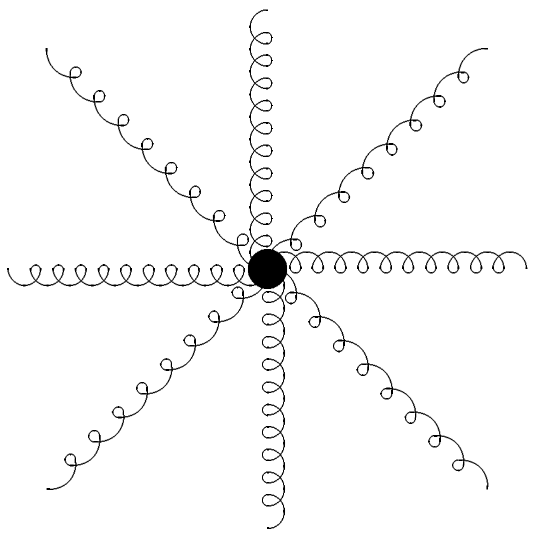
The different styles of gluon are:



The different directions may be illustrated by the following file:

```
% EXAMPLE OF USING FEYNMAN TO DRAW gluon DIAGRAMS IN TEX.
\documentstyle [11pt]{article}
\input FEYNMAN
\begin {document}
\centerline{GLUONBURST using FEYNMAN}
\vskip 0.75in
\hskip 1.5in
\begin{picture}(20000,20000)(-10000,-10000)
\put(0,0){\circle*{1500}}
\drawline\gluon[\N\CENTRAL](0,0)[10]
\drawline\gluon[\NE\REG](0,0)[8]
\drawline\gluon[\E\CENTRAL](0,0)[10]
\drawline\gluon[\SE\REG](0,0)[8]
\drawline\gluon[\S\CENTRAL](0,0)[10]
\drawline\gluon[\SW\REG](0,0)[8]
\drawline\gluon[\W\CENTRAL](0,0)[10]
\drawline\gluon[\NW\REG](0,0)[8]
\end{picture}
```

Which produces:

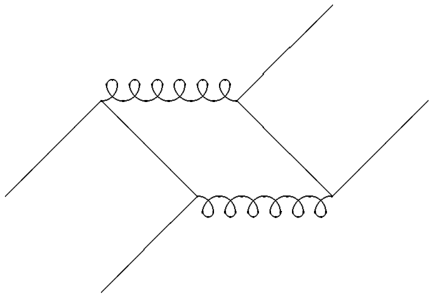


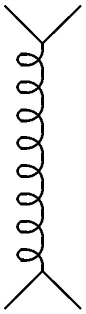
We will see in the next chapter a way of making this look better.

The same points which were noted following the picture *photonburst* in the previous section may be noted here. Again the curvature of the gluons commences in the clockwise sense here and in the counter-clockwise orientation for *flipped* styles. The following examples may prove helpful.

```
\begin{picture}(15000,15000)(-2000,-12000)
\drawline\gluon[\E\FLIPPEDCURLY](0,0)[6]
\drawline\fermion[\NE\REG](\pbackx,\pbacky)[\gluonlengthx]
\drawline\fermion[\SE\REG](\pfrontx,\pfronty)[\gluonlengthx]
\drawline\fermion[\NE\REG](\pbackx,\pbacky)[\gluonlengthx]
\drawline\gluon[\W\FLIPPEDCURLY](\pfrontx,\pfronty)[6]
\drawline\fermion[\SW\REG](\pbackx,\pbacky)[\fermionlength]
\drawline\fermion[\NW\REG](\gluonbackx,\gluonbacky)[\fermionlength]
\drawline\fermion[\SW\REG](\pbackx,\pbacky)[\fermionlength]
\end{picture}
\hskip 3.5in
% Second Picture:
\begin{picture}(6000,15000)
\THICKLINES
\drawline\gluon[\S\REG](3000,14000)[8]
\drawline\fermion[\NE\REG](\pfrontx,\pfronty)[2000]
\drawline\fermion[\NW\REG](\pfrontx,\pfronty)[2000]
\drawline\fermion[\SE\REG](\gluonbackx,\gluonbacky)[2000]
\drawline\fermion[\SW\REG](\gluonbackx,\gluonbacky)[2000]
\end{picture}
```

which renders:





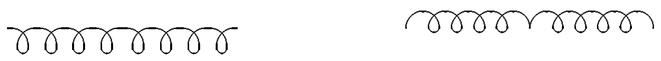
As in the photonic case we use \gluonlengthx,y to draw fermion lines which are the same length as gluon lines. A word of caution. Variables such as '\gluonlengthx' or '\photonlengthy' measure the *displacements* of the line drawn and so may be either positive or negative. That is, \fermionlengthx will be positive for fermions drawn in the E, NE or SE directions and negative for those drawn in the W, NW and SW directions. For N and S fermions it will vanish. The length argument of \drawline, however, must be positive so caution is advised. The '\negate' command described in the section concerning alignments in chapter four is often helpful. The variables *boxlengthx* and *boxlengthy* may also be used. They are generally the absolute values of \particlelengthx and \particlelengthy respectively (however see subsection 2.11.1). These two values must never be altered by the user. The variable '\fermionlength' is the total length of the previously drawn fermion and so is always positive. The second diagram merely demonstrates the \THICKLINES option when used with gluons. For further clarification of these points it may be useful to review the previous section (PHOTONS).

A general result concerning gluon styles is that the \CURLY (\FLIPPEDCURLY) and the \CENTRAL (\FLIPPEDCENTRAL) configurations are frequently more appealing than the \REG (\FLIPPED) configuration if both diagonal and non-diagonal gluons appear in the same diagram. The reason is that slanted (eg. NW) gluons may only be drawn in a \CENTRALGLUON sort of style (but still called \REG) and the \CURLY style is also very similar to this from a spacing and size viewpoint. Another important point is that gluons are by far the most *expensive* sort of lines to draw from the viewpoint of TEX's internal memory (independent of the memory of the driving system). For this reason you will quickly discover that diagrams involving a great many gluons tend to run out of TEX capacity. One way to alleviate the problem is to use LATEX's '\include' facility which allows you to divide up a long program into a series of small ones connected by *auxiliary* files which are automatically produced when LATEX is run. Dozens of photonic, scalar and fermionic diagrams may be constructed for the same memory usage of a single hadronic diagram.

A number a further features exist which will be mentioned here and elaborated on in subsequent chapters. Thus far no multi-gluon vertices have been drawn. Using the techniques presented here it is obvious how this may be done - draw a gluon at (\gluonbackx,\gluonbacky), attaching it at the rear of the last gluon. The results are not always æsthetically appealing so a number a pre-drawn vertices are available. This is the subject of chapter 3. A further feature has to do with attaching gluons end-to-end. This may not seem necessary but is often desirable (see the exercise below). The problem is illustrated by the following file:

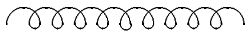
```
\begin{picture}(25000,7000)
\drawline\gluon[\E\REG](0,3000)[4]
\drawline\gluon[\E\REG](\pbackx,\pbacky)[4]

\drawline\gluon[\E\CENTRAL](15000,3000)[4]
\drawline\gluon[\E\CENTRAL](\pbackx,\pbacky)[4]
\end{picture}
```

giving us:

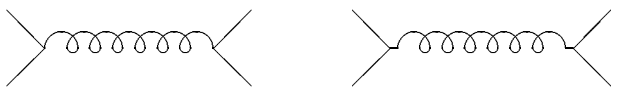


For gluons in the N,S, E and W directions \REG, \CURLY, \FLAT and \SQUASHED configurations may be attached end-to-end trivially. For \CENTRAL gluons and all gluons in the NW, NE, SE and SW directions the connection is imperfect. The `\gluonlink` command, discussed in chapter four, alleviates this:

```
\begin{picture}(10000,5000)
\drawline\gluon[\E\CENTRAL](0,3000)[4]\gluonlink
\drawline\gluon[\E\CENTRAL](\pbackx,\pbacky)[4]
\end{picture}
```

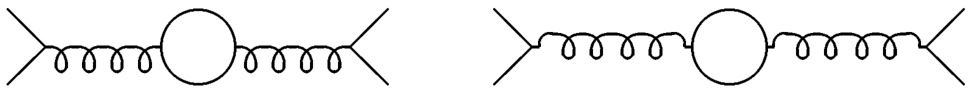


The *stemmed* option demonstrated for photons is also available for gluons and is of significantly more interest, especially for those of the \CENTRAL variety:



UNSTEMMED STEMMED

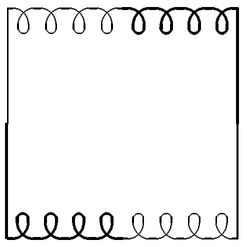
The chief advantage of \CENTRAL gluons is that their endpoints are approximately on the axis along which the line is being drawn. This may be artificially achieved for other styles using the \gluoncap command. The *capped* option is unique to gluons. It is employed to make gluon lines of a non-central variety (*i.e.* those which do not terminate near the axis of the gluon line) appear centralized. An example is:



UNCAPPED CAPPED

There are a number of subtleties involved with the above features which is why their discussion is being postponed. Finally a primitive \drawloop[1] command allows a gluon loop to be drawn. These are also discussed in chapter four.

We conclude with the following exercises. First draw the following which uses \REG style gluons:



How would you draw this using \CENTRAL gluons *without* invoking the \gluonlink command?

[1]currently only available for gluons

2.11 Arrows and Labels

In addition to the actual lines of a Feynman diagram one commonly desires to include labels directly on the diagram, occasionally explanatory text or equations, and quite frequently arrows indicating the flow of momentum or some quantum number. These are accommodated quite easily in **FEYNMAN**.

2.11.1 Arrows

The command for drawing arrows is \drawarrow. The syntax is similar to the \drawline command:

```
\drawarrow[<direction><configuration>]
          (<(x,y) co-ordinates of the arrow>)
```

An arrowhead is produced at the specified (x,y) co-ordinates, generally at the end or middle of a line. The direction is one of the usual compass point directions:

```
\N   \NE   \E   \SE   \S   \SW   \W   \NW
```

If \LDIR or \LINEDIRECTION is used instead then the direction of the last drawn line will automatically be used (pre-supposing that a line has been drawn in the current picture). If the previous line was drawn by the \drawvertex command (see next chapter) then the *initially specified* vertex direction will be used. The *configuration* is one of the following:

```
\ATTIP                    \ATBASE
```

These specify whether the co-ordinates which you've input refer to the position of the **base** of the arrow or the **tip** of the arrow. Arrows may be drawn boldly if \thicklines is in operation. The length of an arrow is given by the variable \arrowlength. When an arrow is attached to the end of a line the \particlelengthx,y are not altered (and neither is \gluonlengthx *etc.*) however the variables *boxlengthx* and *boxlengthy*, mentioned in subsection 2.10.2, assume a different meaning now. The *tip* of the arrow will always have co-ordinates (\ boxlengthx,\ boxlengthy), whether it is drawn with \ATTIP or \ATBASE. The following examples should illustrate these features.

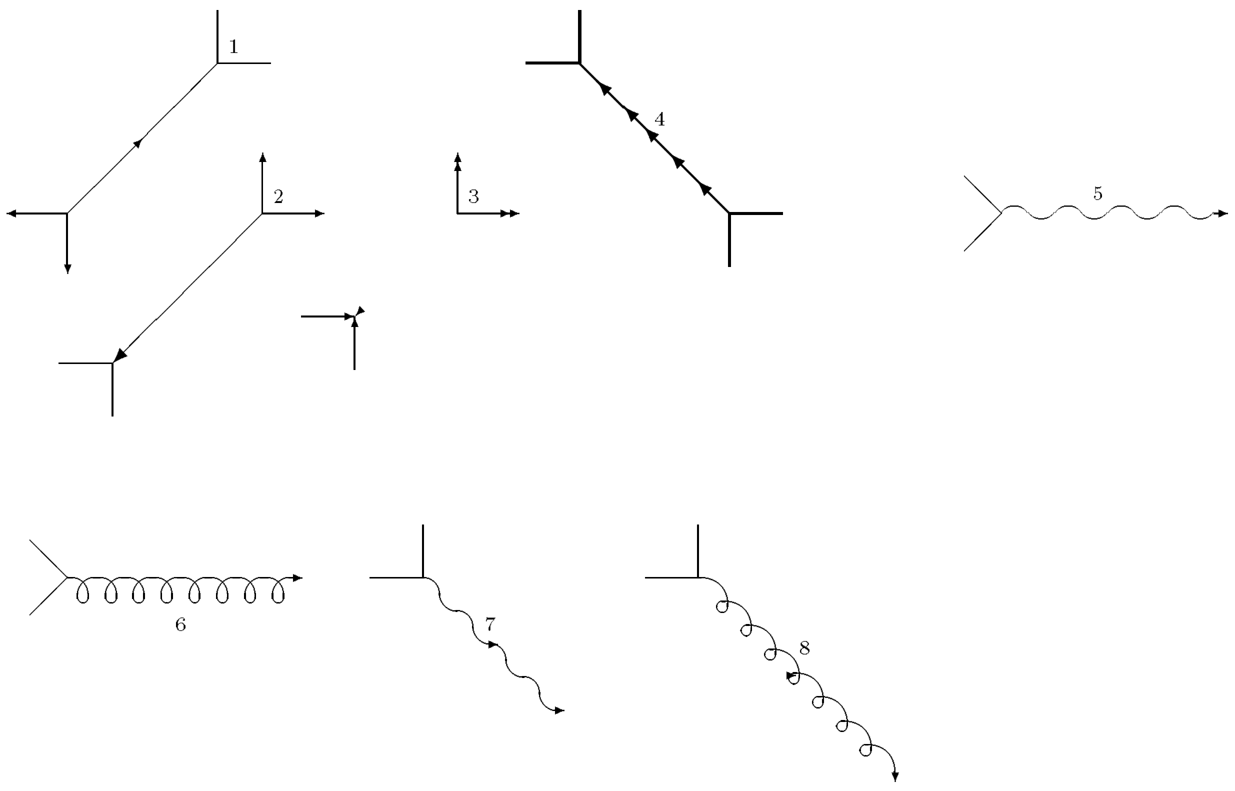
```
\documentstyle [12pt]{article}
\input FEYNMAN
\begin {document}
\bigphotons
\centerline{ARROWS using FEYNMAN.TEX}
\begin{picture}(10000,10000)(0,0)            % PICTURE 1
\drawline\fermion[\NE\REG](0,0)[8000]
\drawarrow[\LDIR\ATTIP](\pmidx,\pmidy)
\drawline\fermion[\E\REG](\fermionbackx,\fermionbacky)[2000]
\drawline\fermion[\N\REG](\fermionfrontx,\fermionfronty)[2000]
\drawline\fermion[\W\REG](0,0)[2000]
```

```
\drawline\fermion[\S\REG](0,0)[2000]
\drawarrow[\S\ATBASE](0,-2000)
\drawarrow[\W\ATBASE](-2000,0)
\end{picture}
\hskip -0.4in
\begin{picture}(10000,10000)(0,0)             % PICTURE 2
\drawline\fermion[\SW\REG](0,0)[8000]
\thicklines
\drawarrow[\SW\ATTIP](\pbackx,\pbacky)
\thinlines
\drawline\fermion[\W\REG](\fermionbackx,\fermionbacky)[2000]
\drawline\fermion[\S\REG](\fermionfrontx,\fermionfronty)[2000]
\drawline\fermion[\E\REG](0,0)[2000]
\drawline\fermion[\N\REG](0,0)[2000]
\drawarrow[\E\ATBASE](2000,0)
\drawarrow[\N\ATBASE](0,2000)
\end{picture}
\hskip -0.4in
\begin{picture}(10000,10000)(0,0)             % PICTURE 3
\drawline\scalar[\SW\REG](0,0)[6]
\drawarrow[\SW\ATTIP](\pbackx,\pbacky)
\drawline\fermion[\E\REG](0,0)[2000]
\drawline\fermion[\N\REG](0,0)[2000]
\drawarrow[\E\ATBASE](2000,0)
\drawarrow[\E\ATTIP](2000,0)
\drawarrow[\N\ATBASE](0,2000)
\drawarrow[\N\ATTIP](0,2000)
\drawline\fermion[\W\REG](\scalarbackx,\scalarbacky)[2000]
\drawline\fermion[\S\REG](\scalarbackx,\scalarbacky)[2000]
\drawarrow[\E\ATTIP](\scalarbackx,\scalarbacky)
\drawarrow[\N\ATTIP](\scalarbackx,\scalarbacky)
\end{picture}
\begin{picture}(10000,10000)(0,0)             % PICTURE 4
\thicklines
\drawline\fermion[\NW\REG](0,0)[8000]
\drawline\fermion[\W\REG](\fermionbackx,\fermionbacky)[2000]
\drawline\fermion[\N\REG](\fermionfrontx,\fermionfronty)[2000]
\drawarrow[\NW\ATBASE](-1000,1000)
\drawarrow[\NW\ATBASE](-2000,2000)
\drawarrow[\NW\ATBASE](-3000,3000)
\drawarrow[\NW\ATTIP](-4000,4000)
\drawarrow[\NW\ATTIP](-5000,5000)
\drawline\fermion[\E\REG](0,0)[2000]
\drawline\fermion[\S\REG](0,0)[2000]
\end{picture}
\begin{picture}(10000,10000)(0,0)             % PICTURE 5
\drawline\photon[\E\REG](0,0)[8]
\drawline\fermion[\E\REG](\pbackx,\pbacky)[200]
\drawarrow[\E\ATBASE](\pbackx,\pbacky)
\drawline\fermion[\SW\REG](0,0)[2000]
\drawline\fermion[\NW\REG](0,0)[2000]
```

```
\end{picture}
\vskip 0.5in
\begin{picture}(10000,10000)(0,0)              % PICTURE 6
\drawline\gluon[\E\REG](0,0)[8]
\drawarrow[\E\ATBASE](\pbackx,\pbacky)
\drawline\fermion[\SW\REG](0,0)[2000]
\drawline\fermion[\NW\REG](0,0)[2000]
\end{picture}
\hskip 0.42in
\begin{picture}(10000,10000)(0,0)              % PICTURE 7
\drawline\photon[\SE\REG](0,0)[8]
\drawarrow[\E\ATBASE](\pmidx,\pmidy)
\drawarrow[\E\ATBASE](\pbackx,\pbacky)
\drawline\fermion[\W\REG](0,0)[2000]
\drawline\fermion[\N\REG](0,0)[2000]
\end{picture}
\begin{picture}(10000,10000)(0,0)              % PICTURE 8
\drawline\gluon[\SE\REG](0,0)[7]
\drawarrow[\E\ATTIP](\pmidx,\pmidy)
\drawarrow[\S\ATBASE](\pbackx,\pbacky)
\drawline\fermion[\W\REG](0,0)[2000]
\drawline\fermion[\N\REG](0,0)[2000]
\end{picture}
```

giving

ARROWS using FEYNMAN.TEX



2.11.2 Labels

We have already seen several examples of how to draw labels. One is to label the overall diagram, at the top or bottom. The \centerline command (see the PHOTONBURST diagram in section 2.9.2) is useful for this.

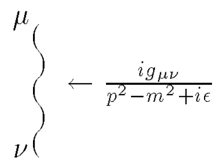
The way to include text within the picture is with the `\put` command. This was done in the sample file in section 1.6. The syntax is:

```
\put(<(x,y) co-ordinates of the lower left point of the text>){<text>}
```

More complicated constructions are possible when used in conjunction with the LaTeX \framebox and \makebox commands (see theLaTeX manual for details). The "text" can be mathematical equations, normal words or even other pictures. As an example:

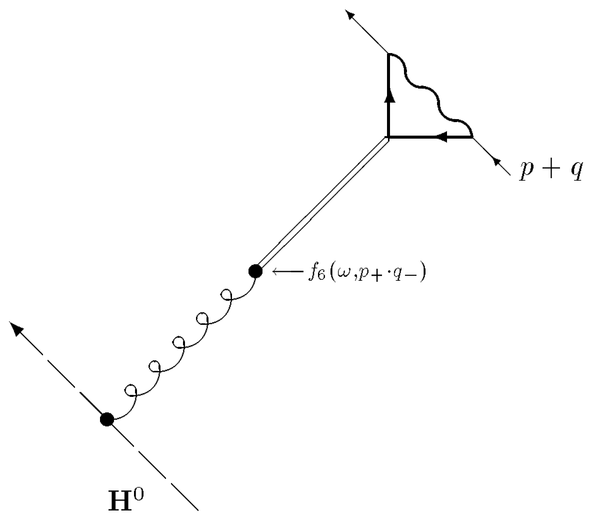
```
\begin{picture}(20000,8000)
\drawline\photon[\N\REG](3000,0)[5]
\put(4000,\pmidy){$\leftarrow \frac{ig_{\mu\nu}}{p^2-m^2+i\epsilon}$}
\put(2000,\pbacky){$\mu$}
\put(2000,\pfronty){$\nu$}
\end{picture}
```

which results in



2.12 Review Exercise for Chapter Two

Produce the following diagram, including labelling. Note the styles and boldness used.



Chapter 3

Drawing Vertices

Most types of vertices will be drawn using the \drawline command. A few specialized vertices, those of three and four gauge bosons, have been pre-defined.

3.1 The \drawvertex Command

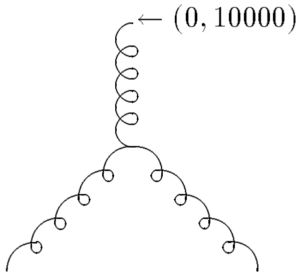
The command *drawvertex* is used to produce a limited number of vertices in conjunction with a number of special options. The syntax is:

```
\drawvertex<particle type>[<direction><number of lines in vertex>]
   (<(x,y) co-ordinates of 'beginning' of vertex>)[length of lines in vertex]
```

where the parameters are similar to those of the \drawline statement. An example is:

```
\drawvertex\gluon[\S 3](0,10000)[4]
```

which would draw a three-gluon vertex commencing from the co-ordinate point (0,10000) (see section 1.5 for a discussion of the co-ordinate grid), with the initial line drawn in a *southerly* direction (towards the bottom of the page) then branching into two other gluons (in the `\SE` and `\SW` direction in this case). Each would be of length four, *i.e.* have four loops. Thus the end result would be a three-gluon vertex centred at some point (which will receive a co-ordinate label (`\vertexmidx,\vertexmidy`)) beneath the point (0,10000). It will have one gluon line running *northward* from that line and terminating at (0,10000) and two others running southeasterly and southwesterly. In point of fact it would be:



We will refer to the initial line drawn (that one specified by the direction parameter) as *line one* and number the lines sequentially in a clockwise sense about the vertex. Therefore we will have line two, three and, possibly, four. In the above example line one is the northerly line, line two is the southeasterly line and line three is the southwesterly line (with respect to the centre). When \drawvertex is executed it returns the following parameters:

`\vertexonex,\vertexoney:`	`The (x,y) co-ordinates of the back of line one.`
`\vertextwox,\vertextwoy:`	`The (x,y) co-ordinates of the back of line two.`
`\vertexthreex,\vertexthreey:`	`The (x,y) co-ordinates of the back of line three.`
`\vertexfourx,\vertexfoury:`	`The (x,y) co-ordinates of the back of line four.`
`\vertexmidx,\vertexmidy:`	`The (x,y) co-ordinates of the middle of the vertex.`
`\vertexcount:`	`The number of vertices printed thus far.`

When drawing a gluon vertex all of the parameters returned by `\drawline\gluon` are defined but take the values of the last gluon drawn (number three for a three-gluon vertex and number four for a four-gluon vertex). Thus when `\drawvertex\gluon[\N 4]...` is encountered \gluonlengthx, \pmidy, \gluonfrontx, *etc.* will be defined with the values appropriate for gluon four (the \E gluon in this instance). The same applies for photonic vertices.

3.2 The Types of Vertex Lines

The kinds of particle lines which **FEYNMAN** can draw with \drawvertex are:

```
\photon
\gluon
```

which are implemented as

```
\drawvertex\photon...
\drawvertex\gluon...
```

Note that this is the only argument of \drawvertex which is in lowercase letters. The \photon option is useful to represent ZWW, ZZWW, graviton vertices and so forth. Drawing $\psi\bar{\psi}\gamma$ and similar vertices has been demonstrated repeatedly. Section 2.1 illustrated the basic command sequence.

3.3 Particle Direction

Each vertex may be drawn in any of eight possible directions, specified by the points of the compass with North always understood as being at the top of the page:

```
\N   \NE   \E   \SE   \S   \SW   \W   \NW
```

This is the second argument of \drawvertex, just as it was for \drawline.

```
\drawvertex\photon[\NW...
```

and so forth. Line one is drawn from the specified point in the indicated direction. The vertex is at the terminus of this line. Note that all directions are in uppercase and, as always, don't omit the backslash. All of the particle lines in the vertex are drawn in these compass directions. Vertices with four particles are drawn in a cross, that is each line is separated by 90°. Vertices with three particle lines are drawn in a 'Y' configuration with the specified direction being the 'base' of the Y. Thus a four-photon vertex drawn in the \NW direction will have line one in the \NW direction, with the vertex at the NW end. Line two will move around clockwise from this by 90° and so will be drawn in the \SW direction out of the hub. Line three will be in the \NW direction and line four in the \NE from the vertex. A three-photon vertex drawn in the \NW direction will again have line one in the same direction. Line two will now be 135° further in a clockwise sense and therefore will be in the \W direction. Line three will be in the \N direction. Lines two and three will always be separated by 90°.

3.4 The Number of Lines

The third parameter will either be a 3 or a 4. It declares whether a three or four particle vertex is to be drawn. Any other entry will result in an error.

3.5 Line Co-ordinate Parameters

3.5.1 Input Parameters

The fourth and fifth arguments of the \drawvertex command are the (x,y) co-ordinates of the *beginning* of particle line number one. These are as measured in *centipoints* on the grid which **FEYNMAN** has established. They are entered in the format (*x co-ordinate*, *y co-ordinate*) where x and y may be integer numbers (between, roughly, -30,000 and +30,000) or variables (counters) with numerical values. A number of variables have been pre-defined and available for use. The user may also define his own (see the section on storing information). Once again some samples may be illustrative:

```
\drawvertex\photon[\SE 3](-1500,12000)[2]
\drawvertex\photon[\E 4](\photonbackx,\photonbacky)[2]
\drawvertex\gluon[\N 4](\Xone,\Yone)[1]
\drawvertex\gluon[\SW 3](3000,\vertexthreey)[7]
```

In the above \photonbackx and \photonbacky are co-ordinates, presumably returned from a previously drawing a photon. \Xone and \Yone are some values stored by the user and \vertexthreey is the ordinate of the endpoint of the third line in the previously drawn vertex.

3.5.2 Output Parameters

In section 3.1 we listed a number of useful positional parameters which are returned when \drawvertex is called. For the appropriate vertices the following are defined, each definition superseding the previous value of the variable.

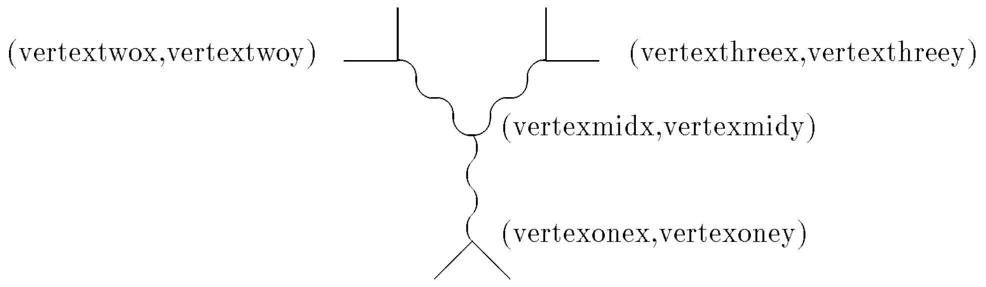
```
\vertexonex,\vertexoney:       The (x,y) co-ordinates of the back of line one.
\vertextwox,\vertextwoy:       The (x,y) co-ordinates of the back of line two.
\vertexthreex,\vertexthreey:   The (x,y) co-ordinates of the back of line three.
\vertexfourx,\vertexfoury:     The (x,y) co-ordinates of the back of line four.
\vertexmidx,\vertexmidy:       The (x,y) co-ordinates of the middle of the vertex.
```

`\vertexcount:                  The number of vertices printed thus far.`

These are graphically illustrated by

```
\begin{picture}(20000,11000)
\drawvertex\photon[\N 3](18500,0)[4]
\drawline\fermion[\SW\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\SE\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\N\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\W\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\N\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\E\REG](\vertexthreex,\vertexthreey)[2000]
\put(\vertexonex,\vertexoney){\quad (vertexonex,vertexoney)}
\put(1000,\vertextwoy){(vertextwox,vertextwoy)}
\put(\fermionbackx,\vertexthreey){\quad (vertexthreex,vertexthreey)}
\put(\vertexmidx,\vertexmidy){\quad (vertexmidx,vertexmidy)}
\end{picture}
```

This yields

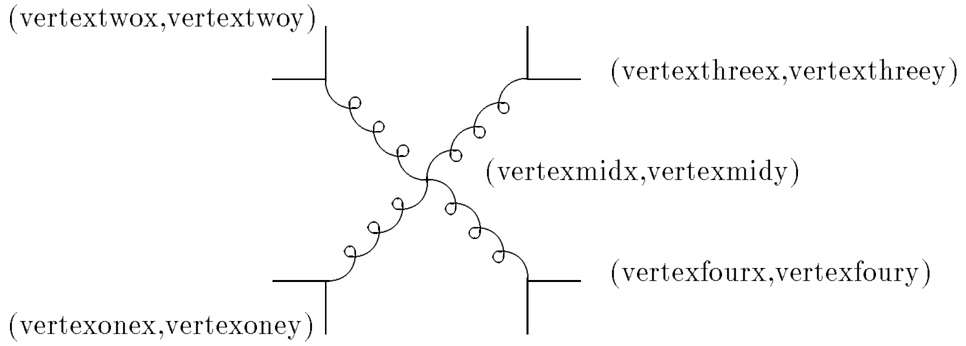


and

```
\begin{picture}(8000,8000)
\drawvertex\gluon[\NE 4](0,0)[3]
\drawline\fermion[\W\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\S\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\N\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\W\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\E\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\N\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\E\REG](\vertexfourx,\vertexfoury)[2000]
\drawline\fermion[\S\REG](\vertexfourx,\vertexfoury)[2000]
\put(\vertexonex,\vertexoney){  (vertexonex,vertexoney)}
\put(\vertextwox,\vertextwoy){  (vertextwox,vertextwoy)}
\put(\vertexthreex,\vertexthreey){  (vertexthreex,vertexthreey)}
\put(\vertexfourx,\vertexfoury){  (vertexfourx,vertexfoury)}
```

```
\put(\vertexmidx,\vertexmidy){  (vertexmidx,vertexmidy)}
\end{picture}
```

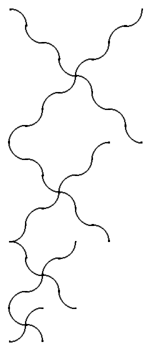
giving us



3.6 Vertex Line Lengths

To make the analogy between \drawline complete, the final parameter to be given to \drawvertex is the *length* of the particles to be drawn. The units in which the length is given varies in way discussed in section 2.6 For gluons the length parameter is the *number of loops*. The actual length of each loop depends upon which style is selected and whether the gluon is drawn diagonally (at a slant) or not. For photons the unit of measure is not a 'wiggle', but a 'half-wiggle'. This enables one to produce a photon which both begins and ends on the 'upward' (or 'downward') part of its oscillation. Each 'leg' of the vertex will be drawn with the requested number of loops or half-wiggles. If a gluon vertex with a different number of loops on each leg were required then the additional loops would be attached subsequently to a basic vertex. Special features are discussed in the next chapter for doing this (\vertexlink and \vertexcap). The particle styles selected will be discussed in sections 3.8 and 3.9.

The following illustrates the length parameter for photons:



which was drawn with

```
\begin{picture}(18000,18000)
\drawvertex\photon[\NE 4](0,0)[1]
\drawvertex\photon[\NE 4](\vertextwox,\vertextwoy)[2]
\drawvertex\photon[\NE 4](\vertextwox,\vertextwoy)[3]
```

```
\drawvertex\photon[\NE 4](\vertextwox,\vertextwoy)[4]
\end{picture}
```

In the above example both \photonlengthx,y and \particlelengthx,y would assume the values of the last photon actually drawn (the \SE photon with four half-wiggles).

3.7 Flipped Vertices

\drawline draws the vertices in a standard configuration. For four-particle vertices all of the lines commence from the hub in a clockwise orientation. For three-particle vertices lines one and two begin in a clockwise curvature and line three in a counter-clockwise sense. It is often very convenient, particularly when linking vertices directly together, to be able to draw one or more lines in a *flipped* configuration, that is flipped about its axis. Obviously this will usually result in an æsthetically unpleasing vertex. **FEYNMAN** admits a few more-or-less appealing flipped vertices to be drawn using the `\flipvertex` command.

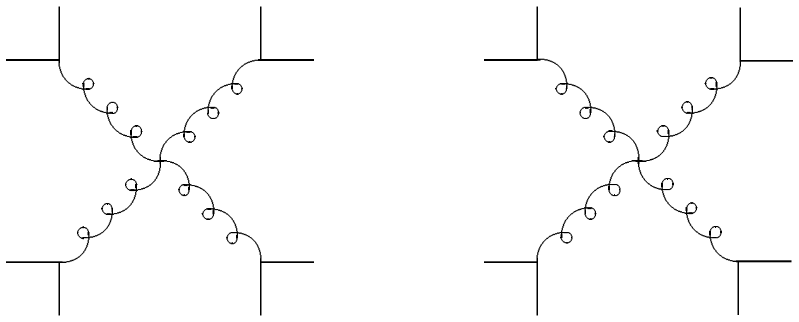
When `\flipvertex` appears directly prior to a \drawvertex statement certain of the lines will be drawn flipped with respect to the default. For four-particle vertices *all* of the lines are flipped. For three-particle vertices only *line one* will be flipped. Other combination may be drawn using \drawline.

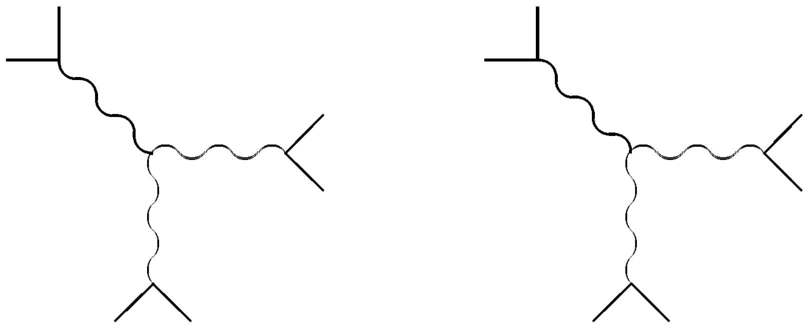
The following file illustrates \flipvertex at work:

```
\begin{picture}(28000,28000)
\drawvertex\gluon[\NE 4](0,19000)[3]
\drawline\fermion[\W\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\S\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\N\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\W\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\E\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\N\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\E\REG](\vertexfourx,\vertexfoury)[2000]
\drawline\fermion[\S\REG](\vertexfourx,\vertexfoury)[2000]

\flipvertex\drawvertex\gluon[\NE 4](18000,19000)[3]
\drawline\fermion[\W\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\S\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\N\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\W\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\E\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\N\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\E\REG](\vertexfourx,\vertexfoury)[2000]
\drawline\fermion[\S\REG](\vertexfourx,\vertexfoury)[2000]

\THICKLINES

\drawvertex\photon[\SE 3](0,8000)[5]
\drawline\fermion[\W\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\N\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\NE\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\SE\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\SW\REG](\vertexthreex,\vertexthreey)[2000]
```

```
\drawline\fermion[\SE\REG](\vertexthreex,\vertexthreey)[2000]

\flipvertex\drawvertex\photon[\SE 3](18000,8000)[5]
\drawline\fermion[\W\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\N\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\NE\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\SE\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\SW\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\SE\REG](\vertexthreex,\vertexthreey)[2000]

\end{picture}
```

which would draw:





\flipvertex may be used with any other options, such as \THICKLINES, as demonstrated above, and features yet to be presented (\linkvertices, \vertexcap *etc.*).

3.8 Photon Vertices

3.8.1 The Anatomy of a Photon Vertex

All four-photon vertices drawn by the \drawvertex command use photons drawn in either \REG or \FLIPPED styles. For three-photon vertices diagonal lines are \CURLY or \FLIPPEDCURLY. The format for drawing a photon vertex with \drawvertex is:

```
\drawvertex\photon[<direction of photon one><number of photons>]
       <(x,y), the co-ordinates of the beginning of the photon one>
       [<number of half-wiggles in each photon line>]
```

The reason for specifying the number of *half*-wiggles is that it is sometimes convenient to have a line begin and end in either orientation (see section 2.6.1 for an example of this).

Photonic vertices may be drawn in any of the eight compass directions. If emboldened photons are desired then \THICKLINES, as opposed to \thicklines, should be used. When the document is [12pt] the \bigphotons statement must be used (as discussed in section 2.9.2).

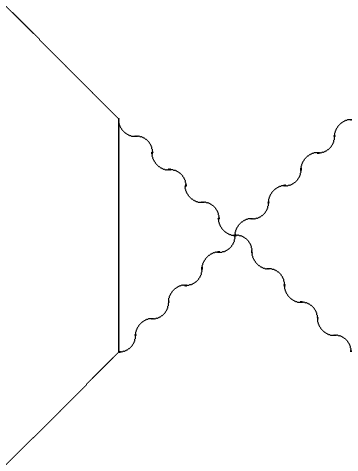
The following parameters are returned by \drawvertex\photon:

```
\photonfrontx,\photonfronty:   The (x,y) co-ordinates of the front of the last line.
\photonbackx,\photonbacky:     The (x,y) co-ordinates of the back of the last line.
\photonlengthx,\photonlengthy: The (x,y) extent of the last line.
\photoncount:                  The number of photons printed thus far.
\pfrontx,\pfronty:             The (x,y) co-ords of the front of the last line.
\pmidx,\pmidy:                 The (x,y) co-ords of the middle of the last line.
\pbackx,\pbacky:               The (x,y) co-ordinates of the back of the last line.
\plengthx,\plengthy:           The (x,y) extent of the line.
\vertexonex,\vertexoney:       The (x,y) co-ordinates of the back of line one.
\vertextwox,\vertextwoy:       The (x,y) co-ordinates of the back of line two.
\vertexthreex,\vertexthreey:   The (x,y) co-ordinates of the back of line three.
\vertexfourx,\vertexfoury:     The (x,y) co-ordinates of the back of line four.
\vertexmidx,\vertexmidy:       The (x,y) co-ordinates of the middle of the vertex.
\vertexcount:                  The number of vertices printed thus far.
```

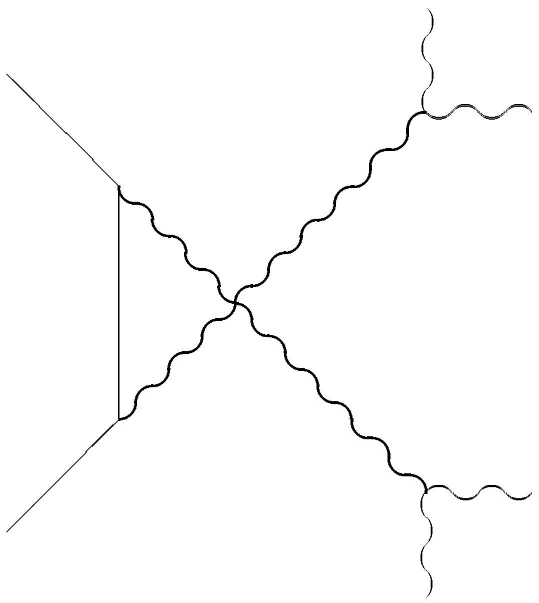
3.8.2 Examples and Details

The following example illustrates that vertices need not only represent branchings but also crossed lines.

```
\begin{picture}(22000,22000)
\drawline\fermion[\NE\REG](0,0)[6000]
\drawvertex\photon[\NE 4](\pbackx,\pbacky)[7]
\drawline\fermion[\N\REG](\vertexonex,\vertexoney)[\photonlengthx]
\drawline\fermion[\N\REG](\fermionbackx,\fermionbacky)[\plengthy]
\drawline\fermion[\NW\REG](\vertextwox,\vertextwoy)[6000]
\end{picture}
```



A couple of items may be noted. In order to draw the connecting fermion line we've drawn it in two stages. We know that \photonlengthx will return the horizontal length of line four of the four-photon vertex. Since line one is the one radiating from the centre in the \SW direction (since it was drawn *to* the centre of the vertex in the \NE direction) we count around clockwise to the fourth line being in the \SE direction. Since the picture is symmetric the 'x' extent of line four is equal to the 'y' extent of lines one and two. Therefore to connect the ends of lines one and two we need to draw a fermion of that length twice. In the second \drawline statement we use \plengthy so that it is the same length as the previous line. If, instead, we'd wanted



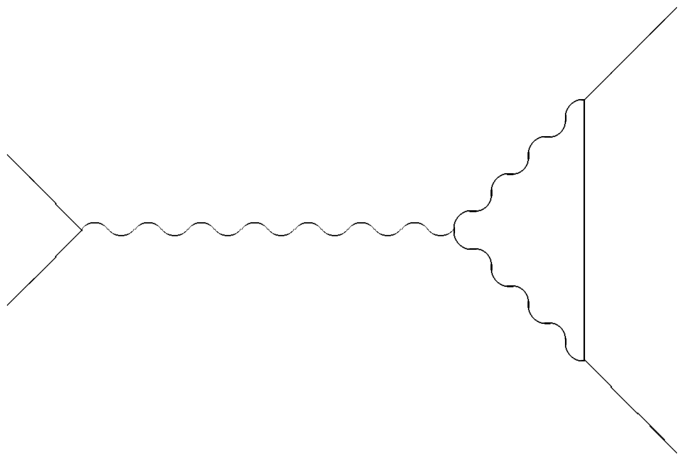
we'd have added

```
\THICKLINES\flipvertex\drawvertex\photon[\NE 3](\vertexthreex,\vertexthreey)[4]
\flipvertex\drawvertex\photon[\SE 3](\vertexfourx,\vertexfoury)[4]
```

and put a `\THICKLINES` modifier before the first \drawvertex and a `\THINLINES` following it. Note how we had to use \flipvertex in order to make the sets of vertices connect properly. In point of fact the above pictures are flawed since the two fermion-fermion-photon vertices are not symmetric. In this instance it would be more appropriate to draw two long photons instead of a vertex.

Photonic vertices may also be *stemmed*, as will be discussed in the next chapter (however see section 2.9 for an example of stemmed photons). Finally we point out, in the form of an exercise, that being able to produce photons with an odd number of half-wiggles again has its uses.

Exercise: Draw the following using \drawvertex. How could you replace the fermion on the right by a scalar?



Note that the diagonal fermion segments on the right have half of the length of the vertical segment.

3.9 Gluon Vertices

3.9.1 The Anatomy of a Gluon Vertex

When a three-gluon vertex is drawn in the N, S, E or W directions using the \drawvertex command, diagonal gluon lines are drawn in a \REG style, while horizontal and vertical lines are drawn in a \CENTRAL configuration. When a three-gluon vertex is drawn in the NE, SW, SE or NW directions using the \drawvertex command, diagonal gluon lines are drawn in a \REG style, while horizontal and vertical lines are drawn in a \CURLY configuration. When preceded by a \flipvertex qualifier line one (and only line one) is flipped to become \FLIPPED, \FLIPPEDCURLY or \FLIPPEDCENTRAL. When a Four-gluon vertex is produced all of the lines will be either \REG (if drawn in a slanted direction) or \CURLY (if in a N, S, E or W direction). When \flipvertex is employed *all* of the lines will be flipped.

The format is:

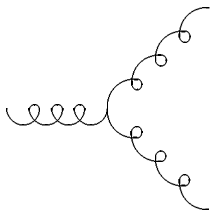
```
\drawvertex\gluon[<direction of gluon one><number of gluons>]
       <(x,y), the co-ordinates of the beginning of the gluon one>
       [<number of loops in each gluon line>]
```

Gluonic vertices may be drawn in any of the eight compass directions. If **bold** gluons are desired then \THICKLINES *or* \thicklines may be used. The following parameters are returned by \drawvertex\gluon:

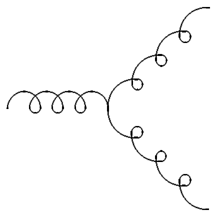
```
\gluonfrontx,\gluonfronty:   The (x,y) co-ordinates of the front of the last line.
\gluonbackx,\gluonbacky:     The (x,y) co-ordinates of the back of the last line.
\gluonlengthx,\gluonlengthy: The (x,y) extent of the last line.
\gluoncount:                 The number of gluons printed thus far.
\pfrontx,\pfronty:           The (x,y) co-ords of the front of the last line.
\pmidx,\pmidy:               The (x,y) co-ords of the middle of the last line.
\pbackx,\pbacky:             The (x,y) co-ordinates of the back of the last line.
\plengthx,\plengthy:         The (x,y) extent of the line.
\vertexonex,\vertexoney:     The (x,y) co-ordinates of the back of line one.
\vertextwox,\vertextwoy:     The (x,y) co-ordinates of the back of line two.
\vertexthreex,\vertexthreey: The (x,y) co-ordinates of the back of line three.
\vertexfourx,\vertexfoury:   The (x,y) co-ordinates of the back of line four.
\vertexmidx,\vertexmidy:     The (x,y) co-ordinates of the middle of the vertex.
\vertexcount:                The number of vertices printed thus far.
```

3.9.2 Examples and Details

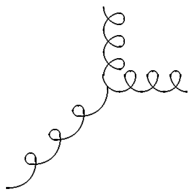
We illustrate the effect of \flipvertex in on three-gluon vertices. The co-ordinate numbering has been done to scale.



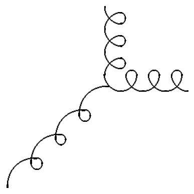
\drawvertex\gluon[\E3](0,30000)[3]



\flipvertex\drawvertex\gluon[\E3](30000,30000)[3]



\drawvertex\gluon[\NE3](5000,10000)[3]

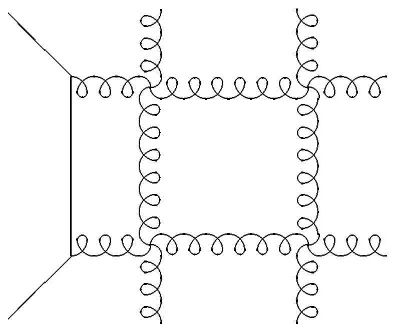


\flipvertex\drawvertex\gluon[\NE3](30000,10000)[3]

The following example illustrates how repeated use of the four-gluon vertex may be employed with great effect. It also illustrates a great shortcoming of using gluon vertices since this small example uses up most of LaTeX's standardly available internal memory. For this reason an *include* statement was used (see LaTeX manual).

```
\begin{picture}(30000,15000)
\drawvertex\gluon[\E 4](15000,0)[3]
\drawline\fermion[\SW\REG](\vertexonex,\vertexoney)[\vertextwoy]
\flipvertex\drawvertex\gluon[\E 4](\vertexthreex,\vertexthreey)[3]
\drawvertex\gluon[\N 4](\vertextwox,\vertextwoy)[3]
\flipvertex\drawvertex\gluon[\W 4](\vertextwox,\vertextwoy)[3]
\drawline\fermion[\NW\REG](\vertexthreex,\vertexthreey)[\fermionlength]
\drawline\fermion[\S\REG](\vertexthreex,\vertexthreey)[\vertexthreey]
\end{picture}
```

giving:

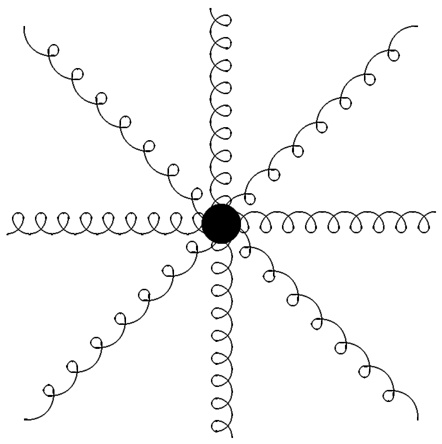


Of note here are the lengths used for the fermions. Since the first vertex is drawn with ordinate $y = 0$, \vertextwoy will be the length of each gluon arm. Thus the lower exterior fermion will also be of this length. The upper fermion is of length \fermionlength, that is the length of the previously drawn fermion, assuring a match. The final fermion line is drawn from the specified point to $y = 0$ using the same trick as above.

An unfortunate defect of **FEYNMAN** may be noticed by returning to the three-gluon vertices on the previous page. Occasionally one might wish to attach the upper left example onto the Eastern line of the lower right example. This would lead to a gluon of \FLIPPEDCURLY (*i.e.* a non-central style) being joined with a \CENTRAL gluon configuration. This cannot be accomplished smoothly without employing arcane tricks usually involving some effort. Similar comments can be made concerning certain combinations of three- and four-gluon vertices.

Vertices may also be linked, capped and stemmed, as will be discussed in the next chapter. Examples of these features were given in section 2.10.2. Also in that section was a diagram labelled "gluonburst". We conclude this chapter by re-creating this using two \drawvertex commands. The reproduction of this diagram is an exercise for the section on placement of information in chapter four. In parting a brief word of caution about TEX's memory usage. In the section on gluon lines it was pointed out that gluons consume vast portions of LATEX's 65500 words of internal memory (only 40000 of which is available to the user). This is trebly true for gluon vertices. Extending the vertices on the following page by two loops per line requires an additional 10000 words. In point of fact this one sample diagram uses 99% of the available memory! For this reason it has been drawn slightly smaller than the example in chapter 2.

GLUONBURST



Chapter 4

Advanced Features

This chapter describes a number of techniques and commands, some alluded to earlier, which may be employed by the user to aid him or her in producing the desired Feynman diagram as closely and quickly as possible.

4.1 Placing It Where You Want It

FEYNMAN has been designed in such a way that particle lines may be drawn in relation with one another with a minimum of effort. Instances invariably arise, however, where the simple commands defined thus far become inadequate to the task. The user might then be condemned to perform some algebra or, much worse, to experiment with the positions of lines and vertices in order to make the appropriate lines connect or place labels in the desired positions.

Fortunately a number of features and tricks are available to reduce this unwanted tedium.

4.1.1 Arithmetic Operations: Manipulating Returned Parameters

The most common placement problems occur when labels are used or closed structures, such as loops, are included. We've seen numerous examples of loops. Those which involve fermions and, to a lesser extent scalars, are most easily constructed. Re-examining the example in section 2.10.2 and the problems posed at the conclusions of sections 2.9.2, 2.10.2, and 2.12 we note one thing in common. Each involves drawing a fermion which is the same (or a simple constant times the) length of a gluon or photon. These make use of the various `\length` parameters which are returned when a line is drawn: `\gluonlengthx`, `\gluonlengthy`, `\photonlengthx`, `\photonlengthy`, `\fermionlengthx`, `\fermionlengthy`, `\scalarlengthx`, and `\scalarlengthy`, as well as `\plengthx` and `\plengthy`, which correspond to the previous line. Each of these has a sign and magnitude depending upon the direction drawn. `\fermionlength` is an especially useful additional measure whose length is always positive and equal to the entire length of the previously drawn fermion. It is often more difficult to prepare loops when the \drawvertex statement has been used since only the length of one line is recorded. This was evident in the example is section 3.8.2.

It is often desirable to know the separation of two arbitrary points, perhaps on two different Feynman diagrams or between points in different sub-branches of a single diagram. Since these points will, in general, be at co-ordinates unknown to the user, the ability to perform simple arithmetic within the TEX file is needed.

TEX defines the following useful commands:

```
\global\advance  <variable name> by <increment  (number or variable)>
\global\multiply <variable name> by <multiplier (number or variable)>
\global\divide   <variable name> by <divisor    (number or variable)>
```

where, as usual, `<,>` are not part of the syntax. In addition **FEYNMAN** provides:

```
\negate       <variable name>
\double       <variable name>
\multroothalf <variable name>
```

In each case the variable name, such as \fermionlength, will be an integer and the result will be an integer (rounded). The first three add, multiply and divide respectively. The second three multiply by minus one, multiply by two, and multiply by $\frac{1}{\sqrt{2}}$ respectively. Each represents an *action*, not a result. Thus to subtract \Y from \X you would enter

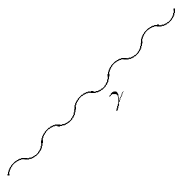
```
\negate\Y
\global\advance \X by \Y
```

but **not**: `\global\advance \X by \negate\Y`. (For the cognoscente you must expand the token "\negate \Y" first). Note that LaTeX defines the commands:

```
\newcounter{<variable name>}
\setcounter{<variable name><variable value}
\addtocounter{<variable name><increment>}
```

but that `\addtocounter` should not be used with **FEYNMAN** unless the variable has been user-defined (see section 4.2) *with \newcounter.*

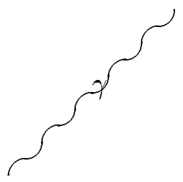
As an example consider labelling the midpoint of a particle line. Suppose that the following is desired:



Where the label is at the midpoint of the photon on the lower side. The *true* (geometric) midpoint is at the co-ordinates (\pmidx,\pmidy). However if we try:

```
\begin{picture}(8000,8000)
\drawline\photon[\NE\REG](0,0)[10]
\put(\pmidx,\pmidy){$\gamma$}
\end{picture}
```

we'd produce:

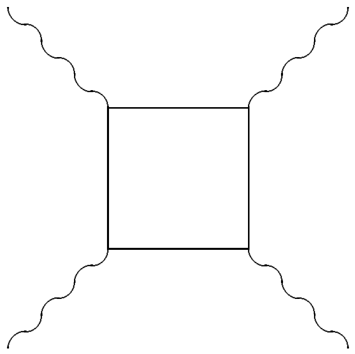


One would like to move the 'γ' about two millimeters to the right and about one and a half millimeters down. This may be readily accomplished with:

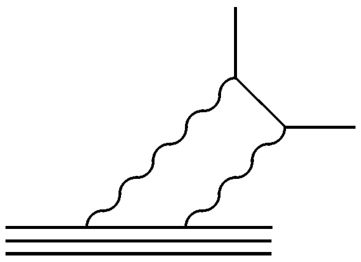
```
\begin{picture}(8000,8000)
\drawline\photon[\NE\REG](0,0)[10]
\global\advance\pmidx by 650
\global\advance\pmidy by -450
\put(\pmidx,\pmidy){$\gamma$}
\end{picture}
```

The exact values used are a question of style and one will frequently experiment to achieve particular æstetics.

A frequent use of \multroothalf occurs when one desires to make a horizontal or vertical line the same length as a slanted line (or vice-versa). As a simple exercise, reproduce the following diagram for Delbrück scattering where all lines are of precisely the same length. Begin by drawing a photon.



As a final example we consider the following case, which is the lowest order QED correction to Coulomb scattering of a lepton from a nucleus which distinguishes between a positron and an electron.



Here the problem is how to space the two photon lines, which differ in length by three half-wiggles, such that both will connect smoothly to the fermions. To facilitate this we introduce two new variables which are returned by \drawline and \drawvertex (and also by \drawloop): *unitboxwidth* and *unitboxheight*. They are (respectively) the width and height (x and y extents) of the **smallest complete unit** of the previously line. For a gluon this is one loop, for a photon an *entire* wiggle, for a scalar one segment plus one gap and for a fermion the entire length. Of course one can equally well use a

```
\global\divide\gluonlengthx by <number of loops>
```

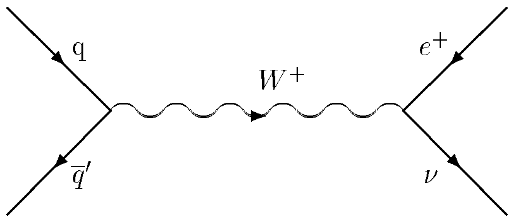
and so forth to obtain the equivalent value. Using these the above may be quickly drawn:

```
\hskip 0.75in
\begin{picture}(10000,10000)(0,0)
\THICKLINES
\drawline\photon[\NE\REG](5000,0)[6]
\multiply\unitboxheight by 3
\multroothalf\unitboxheight
\drawline\fermion[\E\REG](\photonbackx,\photonbacky)[\unitboxheight]
\drawline\fermion[\NW\REG](\photonbackx,\photonbacky)[\fermionlength]
\drawline\fermion[\N\REG](\fermionbackx,\fermionbacky)[\fermionlength]
\drawline\photon[\SW\FLIPPED](\fermionfrontx,\fermionfronty)[9]
\drawline\fermion[\E\REG](\photonbackx,\photonbacky)[7000]
\drawline\fermion[\W\REG](\photonbackx,\photonbacky)[3000]
\drawline\fermion[\E\REG](\fermionbackx,-500)[10000]
\drawline\fermion[\W\REG](\fermionbackx,-1000)[10000]
\end{picture}
```

Since "\particlelengthx", *etc.* represent the *extent*, or increment, of the previous line they are often negative. For a gluon drawn in the \SW direction, for instance, both \gluonlengthx and \gluonlengthy are negative. When using these to compare lengths one must generally employ the "\negate" command. An alternative is to use *boxlengthx* and *boxlengthy* which give the *absolute magnitudes* of the length parameters. There are a few instances, however, where the two might not match (see the sections on stems and caps).

The placement of arrows sometimes requires fine adjustments as well. As an additional exercise try to draw the following as closely as possible. Use a 12-point document size.

Drell-Yan W-Production



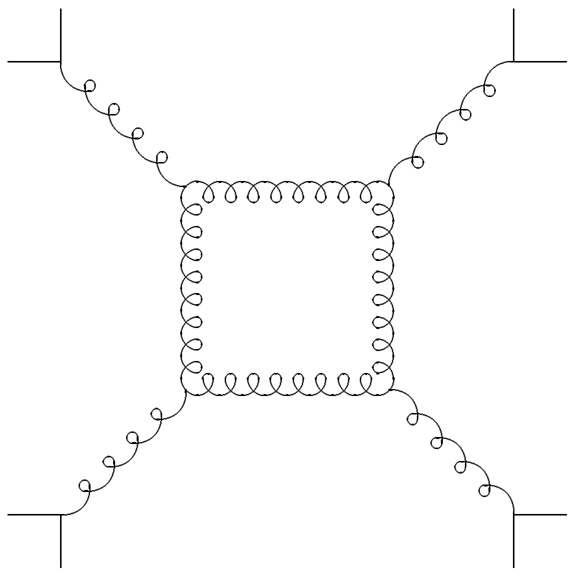
4.1.2 Phantom Commands

It will sometimes happen, generally when vertices and loops are involved, that one wishes that one could draw an invisible line; one which would not actually appear in the final version of a diagram. This is almost always the result of an alignment problem. For instance if, for some reason, you wished to draw two photons, parallel to one another, whose spacing was equal to a *gluon* of five loops in a certain configuration. This might be for an overlay of a transparency. One would like to connect the ends of the photons with a *phantom* gluon.

The solution to this problem is to draw the line in **phantom** mode. Any **FEYNMAN** commands between the statements:

```
\startphantom
      .
      .
      .
\stopphantom
```

are performed but are not printed. In particular all of the returned parameters are evaluated as if the lines and vertices had actually been produced. Note also that **FEYNMAN**'s internal working resources are similarly depleted. The one item that will still be printed while \startphantom is in effect is a labelling command (or any use of \put). \startphantom does not extend beyond the confines of the given picture. The following is a classic example of the use of phantom commands in conjunction with \drawvertex:

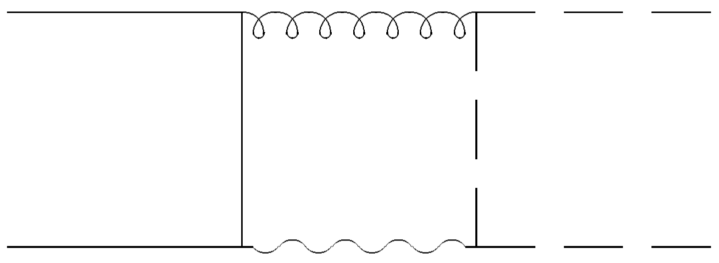


The number of vertices (including phantom vertices) = 35 which is produced by:

```
\begin{picture}(8000,8000)
% This picture isn't printed out.  The \drawvertex statement sets up some
% spacing for the pic below.  It tests the `phantom mode' option.
% The \fermions below are superfluous and merely test \startphantom.
\startphantom
\drawvertex\gluon[\NE 3](0,0)[4]
\drawline\fermion[\SW\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\SE\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\E\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\N\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\E\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\S\REG](\vertexthreex,\vertexthreey)[2000]
\stopphantom
\global\Xtwo=\vertextwox  \global\Ytwo=\vertextwoy
\global\Xthree=\vertexthreex  \global\Ythree=\vertexthreey
\end{picture}
\hskip 2.0in
\begin{picture}(18000,18000)
\global\Xone=\Xthree  \negate\Xone
\global\Yone=\Ytwo    \negate\Yone
\drawvertex\gluon[\NE 3](\Xone,\Yone)[4]
\drawline\fermion[\S\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\W\REG](\vertexonex,\vertexoney)[2000]
\global\Xone=\Xthree  \negate\Xone
\global\Yone=\Ytwo    %\negate\Yone
\drawvertex\gluon[\SE 3](\Xone,\Yone)[4]
\drawline\fermion[\W\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\N\REG](\vertexonex,\vertexoney)[2000]
\global\Xone=\Xthree  %\negate\Yone
\global\Yone=\Ytwo    \negate\Yone
\drawvertex\gluon[\NW 3](\Xone,\Yone)[4]
\drawline\fermion[\E\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\S\REG](\vertexonex,\vertexoney)[2000]
\global\Xone=\Xthree  %\negate\Xone
\global\Yone=\Ytwo    %\negate\Yone
\drawvertex\gluon[\SW 3](\Xone,\Yone)[4]
\drawline\fermion[\E\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\N\REG](\vertexonex,\vertexoney)[2000]
\put(-9500,-14000){The number of vertices (including phantom vertices) =
\number\vertexcount}
\end{picture}
```

We see that the diagram is drawn by reflecting the lower left vertex three times. This is necessary since each vertex can only be drawn beginning at one specific point. First the \NE vertex is drawn, commencing at (0,0), in phantom mode. This provides the x and y displacements of the vertex relative to what will become the centre of the square gluon loop at the co-ordinate (0,0). Then we use reflection symmetry, with the \negate command, to draw all four vertices. The assignment statements such as `\global\Yone=\Ytwo` are discussed below in the section on saving and re-using items.

As an exercise in the use of phantom and arithmetic commands draw the following box diagram for the supersymmetric production process $q\bar{q} \to \tilde{q}\bar{\tilde{q}}$. Make sure that the box is exactly square, that the scalar gaps are precisely one half of the length of the segments, and use a gluon of the \FLAT type.



You might also try labelling the lines. What would be the difficulty in rotating this diagram through 45 degrees? You will find it convenient to use the assignment statement `\global<name>=<name>` mentioned above. Remember that \seglength and \gaplength always resume their defaults after a line is drawn.

4.2 Storing and Re-Using Items

It is often convenient, as well as efficient, to store a line, collection of lines and vertices, or even an entire picture for re-use later. **FEYNMAN** itself continuously stores lines for various periods. It is also useful to be able to store a parameter value for use later in the same or a subsequent picture. Examples of the latter have already been given and will now be presented formally.

4.2.1 Storing and Re-Using Information

The basic command for storing a parameter is

```
\global<parameter 1>=<parameter 2>
```

Several examples were given in the previous section such as:

```
\global\gaplength=\Xone  \global\seglength=\Yone
```

Such statements are particularly useful when some value needs to be altered, such as by

```
\global\advance\particlebackx by 2000
```

and yet also need to be retained for further use. Certain variables, such as `\vertexcount`, are dangerous to adjust and so would need to be copied prior to alteration. Note that `\global<\name>=...` is a TeX primitive operation. As discussed earlier LaTeX has its own equivalent statements but these should not be used in conjunction with variables defined internally by **FEYNMAN**. (For instance attempting to use `\setcounter` in conjunction with `\pmidx` would result in an error.) The `\global` statement prior to `<name>=, \advance` *etc.* is often superfluous but it is safest to use it. This makes the changes instituted common to all subroutines called by **FEYNMAN**. If you continually forget to use it you may someday find the following baffling sort of error:

```
TeX capacity exceeded.
Save Stack overflow.
```

If this occurs go back and insert `\global` just prior to every assignment and alteration statement.

Every variable named must commence with a backslash (\) however one cannot indiscriminantly create a variable, say `\mycopy`, by entering `\global\mycopy=\pbackx`. This is because *control sequences*, such as `\begin`, also commence with backslashes. **FEYNMAN** has predefined a series of empty variable names which may be freely used. These are

`\Xone`	`\Yone`
`\Xtwo`	`\Ytwo`
`\Xthree`	`\Ythree`
`\Xfour`	`\Yfour`
`\Xfive`	`\Yfive`
`\Xsix`	`\Ysix`
`\Xseven`	`\Yseven`
`\Xeight`	`\Yeight`

Not terribly imaginative, I'll grant you, but they work. If the user feels obliged to define his own variable names be warned that there is very little excess LaTeX capacity available for such manœvers. A handfull may be defined. This is done by entering

```
\global\newcount<your variable name>
```

prior to assigning it a value. This may be done on the same line:

```
\global\newcount\pmidycopy   \global\pmidycopy=\pmidy
```

The globals may be omitted (*every* time \pmidycopy is assigned or altered!) if the variable's value is not to be retained from picture to picture. Note, however, that `\pmidycopy` will remain defined as a variable until the end of the program and so `\global\newcount\pmidycopy` must appear only once in the program (this may be prior to the `\begin{document}` statement). To repeat it is to receive an error. Attempts to define variable names such as "\begin" or "\drawline" lead to real trouble.

A word about outputting the value of a variable. As illustrated at the end of the final example of the previous section, one cannot simply say `\put(x,y){\pmidycopy}` and expect the current value of `\pmidycopy` to be placed that the indicated co-ordinates. It must first be converted from a parameter name to what that name represents (*i.e.* evaluated). This may be done by preceeding the variable by \number or \the. The same is true of output to the terminal which is done via the \message command:

```
\message{pmidycopy=\the\pmidycopy}
```

This latter is often useful for debugging your picture when things are not appearing where you thought they would go.

4.2.2 Storing and Re-Using Pictures

FEYNMAN has a number of specialized commands which enables the user to store and re-use one or a number of lines. To store a segment of, or an entire, picture one utilizes LaTeX's *savebox* facility. Unfortunately one will frequently exhaust LaTeX's (tiny) internal working memory when this is attempted. The best solution in this case is to ask a kindly TeX wizard to expand TeX's capacity.

Storing Lines

Why would one wish to store a single particle line? It might be that diagram has the same line repeated over and over commencing from different starting points. The user may simply like a particular line style and length and want to always use it (for instance an \E \FLIPPEDFLAT gluon five loops in length). To store a line under the name `<line name>` the basic command is:

```
\drawandsaveline ‘<line name>’ as <the same arguments as \drawline>
```

Note the ‘...’ around the name (which must commence with a backslash). As an example one could have

```
\drawandsaveline ‘\cutephoton’ as \photon[\S\LONGPHOTON](-2000,500)[7]
```

which would draw a photon as if

```
\drawline\photon[\S\LONGPHOTON](-2000,500)[7]
```

had appeared but save it under the name \cutephoton.

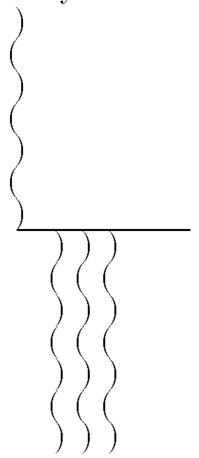
To utilize the saved line one enters

```
\drawoldpic<line name>(x,y)
```

where (x,y) is the co-ordinate to draw the line from. For example

```
\drawandsaveline ‘\cutephoton’ as \photon[\S\LONGPHOTON](-2000,500)[7]
\drawline\fermion[\E\REG](\pbackx,\pbacky)[6500]
\drawoldpic\cutephoton(1500,\pbacky)
\drawoldpic\cutephoton(2500,\pbacky)
\drawoldpic\cutephoton(3500,\pbacky)
```

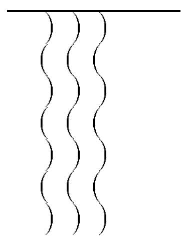
\drawoldpic re-assigns the values of \pfrontx,y; \pmidx,y and \pbackx,y but does **not** alter any other existing paramenters such as \plengthx, \gluonbacky or \photoncount. (\pfrontx,\pfronty) are assigned the (x,y) values and (\pbackx,\pbacky) are incremented by the *current* values of \plengthx and \plengthy. Subsequent usage of \drawoldpic costs no additional internal TEX memory resources. The above example would draw:



The storage space for \cutephoton will be reclaimed following the `\end{picture}` statement. Two other features exist to draw lines. At any time, while in picture mode, the last line drawn via the \drawline command is stored under the name ‘\lastline’ and may be freely used. \lastline is replaced when \drawline is next encountered. Note that you may never use the same name twice when storing a line. To illustrate the use of \lastline, a portion of the previous example could be produced using:

```
\drawline\fermion[\E\REG](0,16000)[6500]
\drawline\photon[\S\LONGPHOTON](1500,\pbacky)[7]
\drawoldpic\lastline(2500,\pfronty)
\drawoldpic\lastline(3500,\pfronty)
```

to wit



One may also store a line *after* it has been drawn by using the *\saveas* command. The statements:

```
\drawline\photon[\S\LONGPHOTON](0,18000)[7]
\saveas '\cutephoton'
```

would be equivalent to the statement

```
\drawandsaveline '\cutephoton' as \photon[\S\LONGPHOTON](0,18000)[7]
```

The latter statement is superior, however, since \saveas actually re-draws the line (in a phantom mode) and thus uses twice as much of TEX's resources. Thus \saveas should be avoided and has been mentioned merely for completeness. Since it saves the last line drawn it is most useful when used subsequent to a \drawvertex statement.

Storing Pictures

It frequently happens that a picture, or part of a picture, needs to replicated. In many instances it is only the labels which need to be changed. The general LATEX command for saving objects is

```
\savebox{<box name>}(<box width>,<box height>)[<position>]{<object>}
```

This saves the 'object' in a 'box' but does not draw it directly. To draw it the \drawoldpic command may again be used as shown below. The box name must begin with a backslash (\). The width and height will be in centipoints. The "position" argument is optional and may be ignored. If so the picture will be centred in the box. Other options are `[tr]` `[br]` `[tl]` `[bl]` which will position the picture in a corner of the named box, and `[t]` `[b]` `[l]` `[r]` which will place the picture along a side. The 'object' in this case is a picture commencing with `\begin{picture}` and terminating with `\end{picture}`. Just as variable names must be initialized so must box names. Prior to the \savebox a command `\global\newsavebox<name>` must appear and this name must not be used again. The saved picture takes the form:

```
\global\newsavebox{\Brehmsstrahlung}
       .
       .
       .
\savebox{\Brehmsstrahlung}(10000,14000)[tl]{
\begin{picture}(10000,140000)(2000,3000)
       .
       .
       .
\end{picture}
}  % end of \savebox
       .
       .
       .
\begin{picture}(18000,22000)
...(various Feynman commands)
\drawoldpic\Brehmsstrahlung(\pbackx,\pbacky)
\global\advance\pbackx by 2000
\drawoldpic\Brehmsstrahlung(\pbackx,\pbacky)
\global\advance\pbackx by 2000
\drawoldpic\Brehmsstrahlung(\pbackx,\pbacky)
...(various Feynman commands)
\end{picture}
```

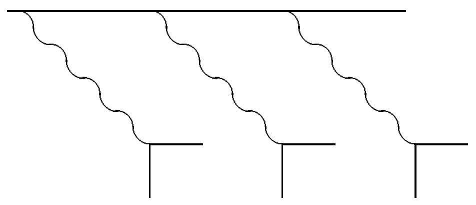
with further possible uses in other diagrams. Of course the picture could be merely stored and directly used as an entire diagram. LaTeX keeps the stored box around thereafter occupying valuable storage space. Thus when one is done with the stored picture the command

```
\sbox{\Brehmsstrahlung}{}
```

will clear the space. The box will continue to exist and new objects may be stored in it without a further \newsavebox statement. One is cautioned that positioning the old picture within a new picture may require some experimentation since the box sizes may not match. The other caution is that TeX's resources tend to become exhausted with a bewildering rapidity which limits the size of stored pictures. The following is a simple example.

```
\global\newsavebox{\ESHOWER}
\savebox{\ESHOWER}(0,0)[tl]{
\begin{picture}(0,0)
\drawline\photon[\SE\REG](0,0)[8]
\drawline\fermion[\E\REG](\photonbackx,\photonbacky)[2000]
\drawline\fermion[\S\REG](\photonbackx,\photonbacky)[2000]
\end{picture}
}   % end savebox
\begin{picture}(15000,5000)
\drawline\fermion[\E\REG](0,5000)[15000]
\drawoldpic\ESHOWER(0,5000)
\drawoldpic\ESHOWER(5000,5000)
\drawoldpic\ESHOWER(10000,5000)
\end{picture}
\sbox{\ESHOWER}{}
```

producing



Note the \sbox used to empty the box. Also note the trick of storing a box of zero dimension. This greatly simplifies the positioning of stored diagram. One might expect that the gluon branches (together with labelling) in the cover diagram were generated this way. Alas there was insufficient memory to store them (gluon vertices are particularly expensive) and they were drawn by duplicating the statements. This saved storage space at the expense of CPU. One may also store boxes within boxes. Thus the entire three-photon diagram above could have been stored and re-used. Indeed this is the basic technique used in constructing lines and vertices.

4.3 Links, Stems and Caps

In this section we discuss a number of important embellishments which may be used to both simplify the construction of a diagram and beautify the results.

4.3.1 Gluon Links

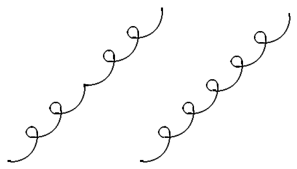
Ideally one would like to draw identical lines (or vertices) following one another and have them join seamlessly, as if a single line had been drawn. For photons, scalars and fermions there is no difficulty in doing this. For many styles of gluon this is likewise true. For diagonally-drawn gluons and those produced in the `\CENTRAL` style this is not the case, as illustrated in section 2.10.2. A similar problem ensues when one wishes to attach a gluon to a vertex constructed from lines drawn in these styles.

Linking Lines

The \gluonlink command is used between two \drawline\gluon statements when the direction of the gluons is `\NW, \NE, \SW` or `\SE` or when the gluons are both in a `\CENTRAL` or `\FLIPPEDCENTRAL` configuration. It's effect is:

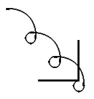
```
\begin{picture}(10000,5000)
% No \gluonlink:
\drawline\gluon[\NE\FLIPPED](0,0)[2]
\drawline\gluon[\NE\FLIPPED](\pbackx,\pbacky)[2]
%
% With \gluonlink:
\drawline\gluon[\NE\FLIPPED](5000,0)[2]\gluonlink
\drawline\gluon[\NE\FLIPPED](\pbackx,\pbacky)[2]
\end{picture}
```

Note that the \gluonlink may also appear on a separate line or preceding the second gluon. There may be labelling statements interspersed but no other lines may be joined. The result is:



Individually its effect is

```
\begin{picture}(5000,5000)
\drawline\gluon[\SE\REG](5000,0)[2]\gluonlink
\drawline\fermion[\N\REG](\gluonbackx,\gluonbacky)[1500]
\drawline\fermion[\SE\REG](\pbackx,\pbacky)[1500]
\end{picture}
```



Linking Vertices

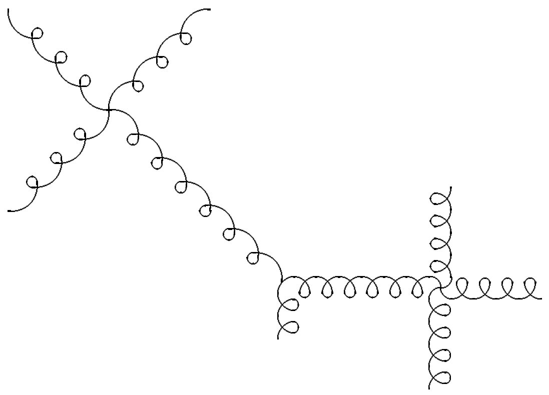
When the number of loops is specified in a \drawvertex command, every line radiating from the center has that number of loops. A \gluonlink following the \drawvertex will pertain solely to the *last* line drawn. In order to permit linkages to other or multiple vertex lines the *\vertexlink* and *\vertexlinks* commands are used. Unlike \gluonlink, \vertexlink(s) *preceds* the \drawvertex statement. The format is `\vertexlink`*n*, where *n* is the line number of the vertex to be linked. For a vertex drawn in a `\S` direction, line one is the first drawn, *i.e.* the \S line drawn *to* the centre (or if you prefer the \N line coming out of the middle of the vertex). The others are counted clockwise around the midpoint `(\vertexmidx,\vertexmidy)`. `\vertexlinks` is an abbreviation of

```
\vertexlink1\vertexlink2\vertexlink3
\vertexlink1\vertexlink2\vertexlink3\vertexlink4
```

for a three and four line vertex respectively. For instance, when the line

```
\vertexlink2\vertexlink3\flipvertex\drawvertex\gluon[\NE 4](0,0)[3]
```

is encountered a four-gluon vertex (four lines each with 3 loops) is drawn in the \flipped configuration. The tip of the southwestern arm is at (0,0) and the northwestern and northeastern arms have links attached (ending at `vertextwox,y` and `vertexthreex,y` respectively). The only time that a `\N, \S, \E or \W` gluon vertex requires a link is for \vertexone when a three-gluon vertex is drawn. Vertices are generally linked to gluon lines but may also be linked together directly as in the following example.



which was produced by the three statements

```
\begin{picture}(8000,12000)
\vertexlink3 \drawvertex\gluon[\SE 4](0,11500)[3]
\flipvertex\drawvertex\gluon[\SE 3](\vertexthreex,\vertexthreey)[2]
\flipvertex\drawvertex\gluon[\E 4](\vertextwox,\vertextwoy)[4]
\end{picture}
```

Note that the \E four-gluon vertex requires no special links (since it is drawn using \CURLY style gluons). In parting note that links may be drawn both with \THINLINES and \THICKLINES and will not appear when used within phantom mode (section 4.1.2).

4.3.2 Stems

Stems are small line segments attached to the end of particle lines. Gluons and photons may be specified with a stem on either end or both ends. Vertices may also be drawn with stems on any or all of the lines.

Stemmed Lines

Examples of stemmed lines have been given in sections 2.9.2 (photons) and 2.10.2 (gluons). They are most æstetically pleasing when used in conjunction with gluons of a centralized style (*i.e.* diagonal gluons or those of styles \CENTRAL and \FLIPPEDCENTRAL). To add a stem to a line enter one of `\frontstemmed` or `\backstemmed` just prior to the \drawline statement. To stem both front and back enter `\stemmed`. That is

```
\frontstemmed\drawline...
\backstemmed\drawline...
\stemmed\drawline...
```

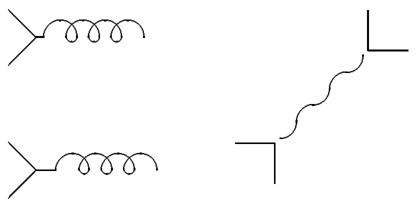
The length of a stem may be set by issuing the command

```
\global\stemlength=<number in centipoints>
```

just prior to the stem command. The length is automatically re-set to its default value. Note that, since diagonal lines of length less than 1415 centipoints cannot be drawn, this is the minimum stemlength possible for a diagonal particle. Attempts to draw a shorter stem will result in a gap. To illustrate

```
\begin{picture}(15000,8000)
\frontstemmed\drawline\gluon[\E\CENTRAL](1000,6000)[3]
\drawline\fermion[\SW\REG](\pfrontx,\pfronty)[1500]
\drawline\fermion[\NW\REG](\pfrontx,\pfronty)[1500]

\global\stemlength=750
\frontstemmed\drawline\gluon[\E\CENTRAL](1000,1000)[3]
\drawline\fermion[\SW\REG](\pfrontx,\pfronty)[1500]
\drawline\fermion[\NW\REG](\pfrontx,\pfronty)[1500]

\stemmed\drawline\photon[\NE\FLIPPED](10000,2000)[5]
\global\Xone=\pbackx  \global\Yone=\pbacky
\drawline\fermion[\S\REG](\pfrontx,\pfronty)[1500]
\drawline\fermion[\W\REG](\pfrontx,\pfronty)[1500]
\drawline\fermion[\N\REG](\Xone,\Yone)[1500]
\drawline\fermion[\E\REG](\Xone,\Yone)[1500]
\end{picture}
```

giving



Stems may be also emboldened via \THICKLINES and will not appear when used in phantom mode. A number of additional features need to be mentioned.

Occasionally it may transpire that the user needs to reference the position of the endpoint of a line both with the stem and without it (perhaps for alignment purposes). After the \stemmed\drawline has been issued the following returned parameters refer to the tips of the *stems*:

```
\pfrontx    \pbackx    \pmidx    \plengthx    \boxlengthx
\pfronty    \pbacky    \pmidy    \plengthy    \boxlengthy
```

whereas the following returned parameters refer to the tips of the *particle*:

```
\gluonfrontx    \gluonbackx    \gluonlengthx
\gluonfronty    \gluonbacky    \gluonlengthy
\photonfrontx   \photonbackx   \photonlengthx
\photonfronty   \photonbacky   \photonlengthy
```

the same is true for \frontstemmed and \backstemmed. This does have a disadvantage in that one frequently must store the values of `\pbackx,\pbacky` *etc.* in order to draw arrows and so forth after intervening \drawline commands.

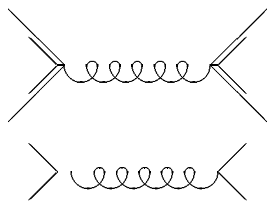
When a stem is drawn it does not truly become part of the line to which it has been attached. For this reason if \drawandsaveline is used with a stem statement the *line* will be saved but *not* the stem(s). That is the statements

```
\global\stemlength=100
\stemmed\drawandsaveline`\littlestem' as \gluon[\W\CENTRALGLUON](0,0)[5]
```

will produce a line as if \drawline had been issued (with stems) but attempts to re-use \littlestem will result in a stemless gluon. The same is true for storing gluons with links and caps attached. Stems saved with the \savebox facility will, of course, be retained. Furthermore in the above example the reduced \stemlength will also revert to its default value. We demonstrate the previous points:

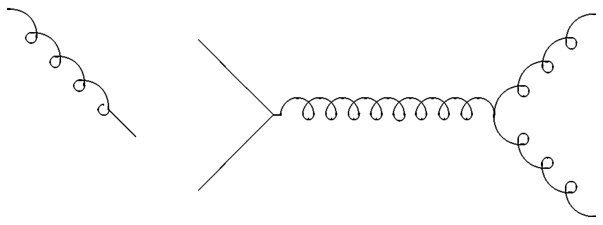
```
\begin{picture}(8000,10000)

\stemmed\drawandsaveline`\unstemmed' as \gluon[\W\CENTRAL](7000,8000)[5]
\global\Xone=\pbackx  \global\Yone=\pbacky
\global\Xthree=\plengthx  \global\Ythree=\plengthy  % store gluon length
\drawline\fermion[\NE\REG](\pfrontx,\pfronty)[1500]
\drawline\fermion[\SE\REG](\pfrontx,\pfronty)[1500]
\drawline\fermion[\NW\REG](\Xone,\Yone)[1500]
\drawline\fermion[\SW\REG](\Xone,\Yone)[1500]

\global\plengthx=\Xthree \global\plengthy=\Ythree   %re-sets gluon lengths

\drawoldpic\unstemmed(7000,4000)
\global\Xone=\pbackx  \global\Yone=\pbacky
\global\Xtwo=\pfrontx  \global\Ytwo=\pfronty
\drawline\fermion[\NE\REG](\gluonfrontx,\gluonfronty)[3000]
\drawline\fermion[\SE\REG](\gluonfrontx,\gluonfronty)[3000]
\drawline\fermion[\NW\REG](\gluonbackx,\gluonbacky)[3000]
\drawline\fermion[\SW\REG](\gluonbackx,\gluonbacky)[3000]
```

```
\drawline\fermion[\NE\REG](\Xtwo,\Ytwo)[1500]
\drawline\fermion[\SE\REG](\Xtwo,\Ytwo)[1500]
\drawline\fermion[\NW\REG](\Xone,\Yone)[1500]
\drawline\fermion[\SW\REG](\Xone,\Yone)[1500]
\end{picture}
```



Note how the line length needed to be stored when \drawoldpic was called. As explained before \drawoldpic re-sets \pfrontx,y to the point from which the line is currently being drawn and calculates \pmidx,y and \pbackx,y based upon that point and the most recent value of \plengthx,y. This is useful when the same line is being repeatedly drawn without intervening \drawline or \drawvertex commands. When such commands occur the lengths are lost. Obviously \plengthx,y were measured inclusive of the stems. To avoid this difficulty one would instead let \Xthree=\gluonlengthx and then assign this value to \plengthx.

The same line may be both stemmed and linked. If this is attempted on the same end you will obtain nonsense.



which comes from

```
\begin{picture}(18000,10000)
\global\stemlength=1416
\backstemmed\drawline\gluon[\SE\REG](0,8000)[3]\gluonlink

\frontstemmed\drawline\gluon[\E\CENTRAL](10000,4000)[4]\gluonlink
\drawline\fermion[\NW\REG](\pfrontx,\pfronty)[4000]
\drawline\fermion[\SW\REG](\pfrontx,\pfronty)[4000]
\flipvertex\drawvertex\gluon[\E 3](\gluonbackx,\gluonbacky)[3]
\end{picture}
```

Stemmed Vertices

The *\stemvertex* and *\stemvertices* qualifiers add a stem or stems to the rear of vertex lines drawn with a \drawvertex command. The syntax is:

```
\stemvertex<n>\drawvertex...
\stemvertices\drawvertex...
```

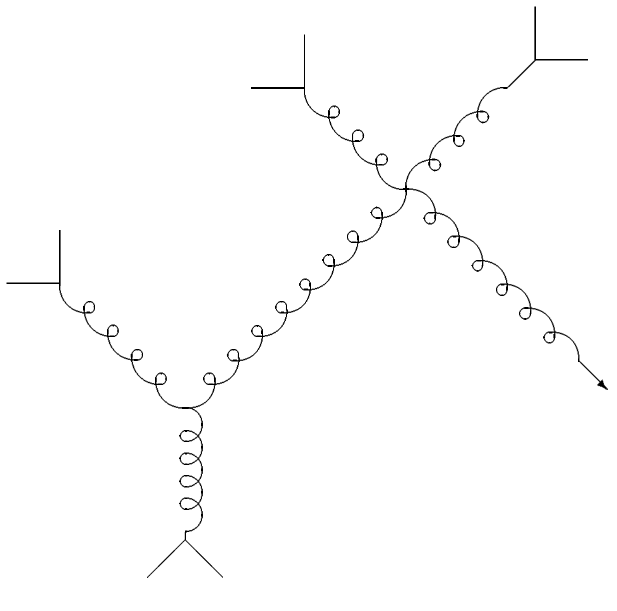
where `<n>` is the number of the vertex line to be stemmed. \stemvertices stems all of the lines. Theses commands remain in effect only for the current \drawvertex. For example

```
\stemvertex2\drawvertex\photon[\E 3](\Xone,\Yone)[6]
\global\stemlength=1415
\stemvertices\drawvertex\gluon[\SW 4](0,0)[4]
```

will draw a three-photon vertex with a stem of standard length on the second line (in the NE direction) and a four-gluon vertex with stems of length 1415 on each of the four lines. Note that the stemlength is not reset until the entire vertex has been completed. To stem just lines two and four one would have written

```
\global\stemlength=1415
\stemvertex2\stemvertex4\drawvertex\gluon[\SW 4](0,0)[4]
```

and so forth. In this case \vertexonex, \vertexoney *etc.* refer to the end of the stem and there is no easy way of marking the end of a given line *sans stem.* \stemvertex may be used in conjunction with \vertexlink and \vertexcap. As an exercise try to produce:



where the diagonal stems are of length 1500. Use links and stems wherever possible.

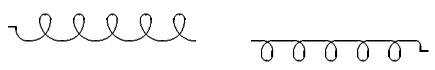
4.3.3 Capped Gluons

Gluons drawn in the styles \REG, \FLIPPED, \CURLY, \FLIPPEDCURLY, \FLAT, \FLIPPEDFLAT and \SQUASHED commence and conclude on one *edge* of the particle line as opposed to \CENTRAL gluons which begin and end near the middle of the line. This can be seen clearly in section 2.10.1. On occasion one wishes to draw a gluon which is central on one side and non-central and the other. To centralize a gluon the *\gluoncap* modifier is invoked. Similarly gluon verticies may be capped via *\vertexcap* and *\vertexcaps*.

Capped Lines

The *\gluoncap* statement is similar to the \gluonlink statement. Only the rear of the gluon is ever capped. If both ends need to be centralized it is best to use a \CENTRAL gluon. Any horizontal or vertical gluon may be capped but \CENTRAL and \FLIPPEDCENTRAL gluons will receive caps of size zero. Diagonal gluons may be stemmed but not capped. Examples are:

```
\drawline\gluon[\W\FLAT](8000,0)[5]\gluoncap
\drawline\gluon[\E\SQUASHED](10000,0)[5]\gluoncap
```



Caps may be emboldened and are not printed in phantom mode. The length of the stem of the cap is \stemlength and this may be re-set as described in the section on stems (4.3.2). Unlike the case of links both \gluonbackx,y and \pbackx,y will refer to the end of the cap. The details of the uncapped gluon may be stored prior to issuing the \gluoncap command. (and ditto for \gluonlengthx,y *etc.*). As an exercise try to reproduce the capped example at the conclusion of section 2.10.2 (note that this picture is better drawn using stemmed CENTRAL gluons).

Capped Vertices

Gluon caps find their greatest use when appended to \drawvertex-generated vertices. One does not have a choice of which gluon type is produced when \drawvertex\gluon is employed. Capped vertices may be seen in the lower left of the cover illustration. In this case gluons of types \CURLY and \FLIPPEDCURLY are capped. The syntax is

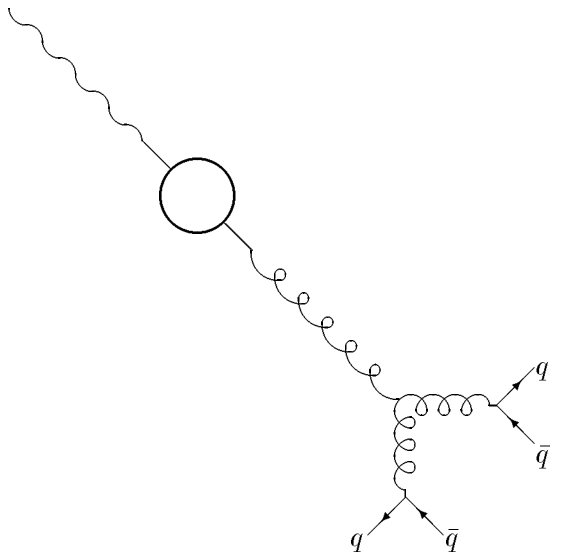
```
\vertexcap<n>\drawvertex\gluon...
\vertexcaps\drawvertex\gluon...
```

Where, once again, `<n>` is the gluon line to be capped. \vertexcaps caps all lines which can be capped. Examples would be

```
\vertexcap1\vertexcap4\drawvertex\gluon[\S 4](0,0)[6]
\vertexcaps\drawvertex\gluon[\W 3](0,0)[3]
```

In the latter case no caps would be drawn since all three lines are of a CENTRAL variety. \vertexcap may be used in conjunction with \vertexlink and \stemvertex.

Exercise: Produce



In the above the circle is of diameter $1000\sqrt{2}$ and the 'stems' attached to it of length 1500. Try to work from the upper left to the lower right.

4.4 Loops

There is a primitive facility in **FEYNMAN** for drawing loops. In the current version it is only capable of drawing gluon loops. Fermion loops may be drawn with the \circle command as illustrated, for instance, in the exercise at the conclusion of the previous section. In future versions it is hoped that photonic (and perhaps scalar) loops may be available.

4.4.1 Non-central Loops

To draw a circular gluonic loop, or portion thereof, one uses the *drawloop* command whose syntax is reminiscent of the \drawline and \drawvertex commands.

```
\drawloop<particle type>[<initial direction><extent>](x,y)
```

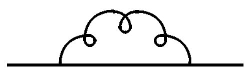
where the particle type is currently limited to **gluon**. The 'extent' is the number of eighths of a complete loop which are to be drawn (1-8) with 8 indicating a complete, closed, circular loop. The loop commences from the point (x, y) in the direction of 'initial direction' and continues being drawn clockwise until the requested number of eighths have been completed. A number of useful parameters are returned after \drawloop has been called.

```
\loopfrontx,\loopfronty   co-ords of beginning point of loop
\loopbackx,\loopbacky     co-ords of opposite point of loop
\loopmidx,\loopmidy       co-ords of geometric middle point of loop
\pbackx,\pbacky           co-ords of end point of loop
\gluonbackx,\gluonbacky   co-ords of end point of loop (for a gluon loop)
```

It must be noted that \loopbackx,y and \loopmidx,y are *only* assigned values if at least half of a loop has been drawn, that is if the loops 'extent' is at least 4. Some examples will illustrate this:

```
\THICKLINES
\begin{picture}(8000,8000)
\drawline\fermion[\E\REG](0,0)[2000]
\drawloop\gluon[\NE 3](\pbackx,\pbacky)
\drawline\fermion[\E\REG](\pbackx,\pbacky)[2000]
\drawline\fermion[\W\REG](\pbackx,\pbacky)[7000]
\end{picture}
```

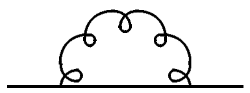
would produce



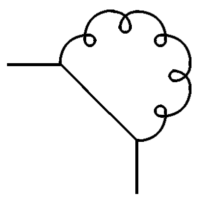
whereas

```
\THICKLINES
\begin{picture}(8000,8000)
\drawline\fermion[\E\REG](0,0)[2000]
\drawloop\gluon[\N 5](\pbackx,\pbacky)
\drawline\fermion[\E\REG](\pbackx,\pbacky)[2000]
\drawline\fermion[\W\REG](\pbackx,\pbacky)[7000]
\end{picture}
```

would give



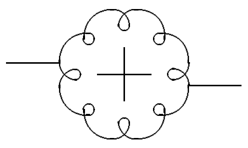
As can be seen \THICKLINES works. So does phantom mode. To produce:



one would enter

```
\begin{picture}(8000,8000)
\drawline\fermion[\E\REG](0,0)[2000]
\drawloop\gluon[\NE 5](\pbackx,\pbacky)
\negate\gluonbackx
\global\advance\fermionbackx by \gluonbackx
\double\fermionbackx   \multroothalf\fermionbackx
\drawline\fermion[\NW\REG](\pbackx,\pbacky)[\fermionbackx]
\drawline\fermion[\S\REG](\pfrontx,\pfronty)[2000]
\end{picture}
```

Finally, to point out exactly where \loopmidx,y and \loopbackx,y are, we try:



created via

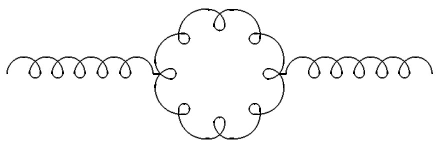
```
\drawloop\gluon[\NE 8](0,0)

\drawline\fermion[\W\REG](\loopfrontx,\loopfronty)[2000]
\drawline\fermion[\E\REG](\loopbackx,\loopbacky)[2000]
\drawline\fermion[\E\REG](\loopmidx,\loopmidy)[1000]
\drawline\fermion[\W\REG](\loopmidx,\loopmidy)[1000]
\drawline\fermion[\S\REG](\loopmidx,\loopmidy)[1000]
\drawline\fermion[\N\REG](\loopmidx,\loopmidy)[1000]
\drawline\fermion[\S\REG](\pbackx,\pbacky)[2000]
```

4.4.2 Central Loops

As can be seen from the previous example, \loopfrontx,y and \loopbackx,y are not at the same level as \loopmidx,y or each other. This is because the peculiar way in which LaTeX draws gluons. The result is that certain loops may appear stilted, for instance the gluon loop correction to a gluon propagator. To alleviate this problem an alternate mode exists. If the user requests that the number of 'eighths' drawn be zero then a complete loop will be drawn in the *central* mode. In this mode the only parameters to be entered are the co-ordinates of the **centre** of the loop. The initial direction may be specified as anything and is irrelevant. ONLY complete loops may be drawn in this fashion. The returned parameters are the same but have slightly altered interpretations. \loopmidx,y are as before. \loopfrontx,y refer to the *left-most* point (that furthest west) of the loop and \loopbackx,y refers to the *right-most* point (furthest east). For instance

```
\drawloop\gluon[\NE 0](0,0)
```

would produce a complete loop centred at (0,0) with \loopfronty and \loopbacky both equal to zero. In this way one could produce



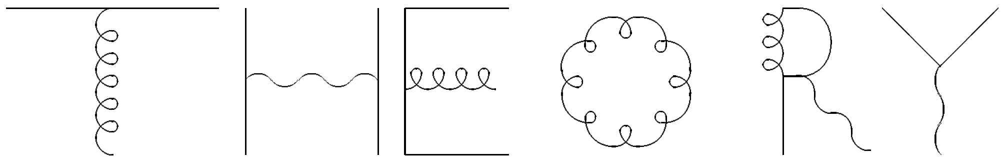
with the three statements

```
\drawloop\gluon[\NE 0](0,0)
\frontstemmed\drawline\gluon[\E\CENTRAL](\loopbackx,\loopbacky)[5]
\frontstemmed\drawline\gluon[\W\FLIPPEDCENTRAL](\loopfrontx,\loopfronty)[5]
```

4.5 Review Exercises for Chapter 4

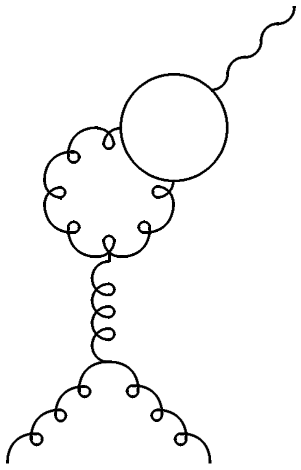
A:

Describe (but don't reproduce) how you would make the following example of 'Feyn Art':



B:

Reproduce the 'balloon in the tree' diagram:



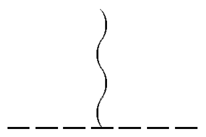
Appendix A

Solution to Exercises

It should first be noted that none of these solutions is unique. There are frequently many ways of drawing a diagram with **FEYNMAN**, the number increasing dramatically with the complexity. The following are reasonable (although not necessarily optimal) solutions to all of the problems posed throughout the manual.

A.1 Exercises for section 2.9.2

The question was posed as to why one cannot draw a scalar in two pieces, commencing from the same point but in opposite directions. The answer is that all scalars both *begin* and *end* with a line segment. Hence the central segment of such a scalar would be twice the length of the surrounding segments. It is easily to overcome this difficulty by drawing the central segment separately, either as a fermion or as a scalar of extent [1], and then drawing the two scalars from the ends. For example to draw



You could enter:

```
\begin{picture}(15000,5000)
\drawline\photon[\N\CURLY](7000,0)[4]
\drawline\fermion[\W\REG](7000,0)[400]
\drawline\fermion[\E\REG](7000,0)[400]
\seglength=800  \gaplength=250   % It is better to use \global\seglength=800 etc.
\drawline\scalar[\W\REG](7400,0)[4]
\seglength=800  \gaplength=250
\drawline\scalar[\E\REG](6600,0)[4]
\end{picture}
```

The diagram at the conclusion of 2.9.2 may be produced by the following eight commands (or permutations thereof):

```
\begin{picture}(20000,20000)
\thicklines\drawline\photon[\N\FLIPPEDCURLY](3000,3000)[7]
\drawline\fermion[\NW\REG](\pbackx,\pbacky)[\photonlengthy]
\drawline\fermion[\E\REG](\fermionfrontx,\fermionfronty)[\fermionlength]
```

```
\drawline\fermion[\SW\REG](\photonfrontx,\photonfronty)[\photonlengthy]
\drawline\fermion[\E\REG](\photonfrontx,\photonfronty)[\photonlengthy]
\drawline\fermion[\N\REG](\pbackx,\pbacky)[\photonlengthy]
\drawline\photon[\SE\REG](\fermionfrontx,\fermionfronty)[7]
\drawline\photon[\NE\FLIPPED](\fermionbackx,\fermionbacky)[7]
\end{picture}
```

A.2 Exercises for section 2.10.2

```
\begin{picture}(20000,20000)
\THINLINES   % Upper-left corner:
\drawline\gluon[\E\REG](18000,18000)[4]
\drawline\fermion[\S\REG](\pfrontx,\pfronty)[\gluonlengthx]
\THICKLINES  % Upper-right corner:
\drawline\gluon[\E\REG](\gluonbackx,\gluonbacky)[4]
\drawline\fermion[\S\REG](\pbackx,\pbacky)[\gluonlengthx]
\THINLINES   % Lower-right corner:
\drawline\fermion[\S\REG](\pbackx,\pbacky)[\gluonlengthx]
\drawline\gluon[\W\REG](\fermionbackx,\fermionbacky)[4]
\THICKLINES  % Lower-left corner:
\drawline\gluon[\W\REG](\gluonbackx,\gluonbacky)[4]
\drawline\fermion[\N\REG](\pbackx,\pbacky)[\fermionlength]
\end{picture}
```

To draw this with \CENTRAL gluons one could divide each side into *three* pieces, not two. Alternatively, starting at one point draw the \THICKLINES segment and then starting *at the same point* draw the \THINLINES piece on top of it. For example:

```
\THICKLINES\drawline\gluon[\E\CENTRAL](0,0)[4]
\THINLINES\drawline\gluon[\E\CENTRAL](0,0)[9]
```

A.3 Exercise for section 2.12

```
\begin{picture}(20000,20000)   % Start at upper right.
\thicklines\drawline\photon[\SE\REG](18000,18000)[5]
\drawline\fermion[\S\REG](\photonfrontx,\photonfronty)[\boxlengthy]
\drawarrow[\N\ATBASE](\pmidx,\pmidy)
\drawline\fermion[\W\REG](\photonbackx,\photonbacky)[\photonlengthx]
\drawarrow[\W\ATTIP](\pmidx,\pmidy)
% Draw 2 small lines to connect the vector meson (parallel lines) at a corner:
\thinlines\drawline\fermion[\S\REG](\fermionbackx,\fermionbacky)[150]
\drawline\fermion[\W\REG](\fermionfrontx,\fermionfronty)[150]
\drawline\fermion[\SW\REG](\fermionbackx,\fermionbacky)[7000]  % Upper || line
\drawline\fermion[\S\REG](\fermionbackx,\fermionbacky)[75]  % Find the centre of
\drawline\fermion[\E\REG](\fermionbackx,\fermionbacky)[75]  % the double fermion
```

```
\drawline\gluon[\SW\REG](\fermionbackx,\fermionbacky)[5]
\put(\gluonfrontx,\gluonfronty){\circle*{500}} % A ‘blob’: the vector->gauge trans.
\drawline\fermion[\S\REG](\fermionbackx,\fermionbacky)[75]  % Draw to position the
\drawline\fermion[\E\REG](\fermionbackx,\fermionbacky)[75]  % Second parallel line
\drawline\fermion[\NE\REG](\fermionbackx,\fermionbacky)[7000]  % Lower || line
\put(\gluonbackx,\gluonbacky){\circle*{500}}
\gaplength=300
\drawline\scalar[\NW\REG](\gluonbackx,\gluonbacky)[3]  % Left half of scalar
\thicklines\drawarrow[\NW\ATBASE](\scalarbackx,\scalarbacky)
\gaplength=300   \thinlines
\drawline\scalar[\SE\REG](\gluonbackx,\gluonbacky)[3]  % Right half of scalar
\put(\pfrontx,\scalarbacky){{\bf H}$^0$}
\put(\gluonfrontx,\gluonfronty){\ $\,{}_{\longleftarrow f_6(\omega,p_+\cdot q_-)}$}
\drawline\fermion[\NW\REG](\photonfrontx,\photonfronty)[2000]
\drawarrow[\NW\ATBASE](\pbackx,\pbacky)
\drawline\fermion[\SE\REG](\photonbackx,\photonbacky)[2000]
\drawarrow[\NW\ATTIP](\pmidx,\pmidy)
\put(\pbackx,\pbacky){$\,\,p+q$}   % \, gives extra space in math mode.
\end{picture}
```

A.4 Exercise for section 3.8

```
\begin{picture}(20000,12000)(0,6000)
\drawline\photon[\E\REG](4000,0)[7]   % Left half of long photon.
\drawline\fermion[\NW\REG](\photonfrontx,\photonfronty)[4000]
\drawline\fermion[\SW\REG](\photonfrontx,\photonfronty)[4000]
\drawvertex\photon[\E 3](\photonbackx,\photonbacky)[7]  % Continue long photon.
\drawline\fermion[\S\REG](\vertextwox,\vertextwoy)[\vertextwoy]  % Upper half
\drawline\fermion[\N\REG](\vertexthreex,\vertexthreey)[\vertextwoy] % Lower half
\drawline\fermion[\NE\REG](\vertextwox,\vertextwoy)[\vertextwoy]
\drawline\fermion[\SE\REG](\vertexthreex,\vertexthreey)[\vertextwoy]
\end{picture}
```

Since we've slyly selected the main axis to have zero ordinate (y=0) that we can use \vertextwoy as a length in order to ensure that the diagonal fermion segments on the right have half of the length of the vertical segment. The vertical fermion is drawn in two sections.

In order to draw a scalar in place of the fermion line on the right we must set \seglength and \gaplength so that the vertical line will connect properly with the photons. The section on spacing in chapter four shows how to do this using the \phantom commands. Here we could just divide the line length by an integer. For instance the commands

```
\global\divide \vertextwoy by 4
\global\gaplength=\vertextwoy
\global\multiply\vertextwoy by 2
\global\seglength=\vertextwoy
\drawline\scalar[\S\REG](\vertextwox,\vertextwoy)[3]
```

would draw a vertical connecting scalar with three segments.

A.5 Exercises for section 4.1.1

```
% Exercise 1: Delbruck Scattering:

\hskip 0.75in                    % Move the whole picture to the right.
\begin{picture}(10000,10000)(0,0)
\drawline\photon[\SE\FLIPPED](0,5000)[6]   % Start from the upper left.
\double\photonlengthx  \multroothalf\photonlengthx  % mults by root 2.
\drawline\fermion[\E\REG](\photonbackx,\photonbacky)[\photonlengthx]
\drawline\photon[\NE\REG](\pbackx,\pbacky)[6]
\drawline\fermion[\S\REG](\photonfrontx,\photonfronty)[\fermionlength]
\drawline\photon[\SE\FLIPPED](\pbackx,\pbacky)[6]
\drawline\fermion[\W\REG](\photonfrontx,\photonfronty)[\fermionlength]
\drawline\photon[\SW\REG](\pbackx,\pbacky)[6]
\drawline\fermion[\N\REG](\photonfrontx,\photonfronty)[\fermionlength]
\end{picture}


% Exercise 2:  Drell-Yan

\documentstyle [12pt]{report}
\begin{document}
\input FEYNMAN
\begin{center}    %  Everything to be centred.
Drell-Yan W-Production

\begin{picture}(10000,10000)    % Note that the 10000 x 10000 box is centred.
\bigphotons   % needed in 12-pt.
\THICKLINES
\drawline\photon[\E\REG](5000,5000)[11]
\drawarrow[\E\ATBASE](\pmidx,4820)
\put(\pmidx,5800){$W^+$}
\drawline\fermion[\NW\REG](\photonfrontx,\photonfronty)[5500]
\drawarrow[\SE\ATBASE](\pmidx,\pmidy)
\put(3500,7100){q}
\drawline\fermion[\SW\REG](\photonfrontx,\photonfronty)[5500]
\drawarrow[\SW\ATBASE](\pmidx,\pmidy)
\put(3500,2300){$\overline{q}'$}
\drawline\fermion[\NE\REG](\photonbackx,\photonbacky)[5500]
\drawarrow[\SW\ATBASE](\pmidx,\pmidy)
\global\advance \pmidx by -1400
\put(\pmidx,7100){$e^+$}
\drawline\fermion[\SE\REG](\photonbackx,\photonbacky)[5500]
\drawarrow[\SE\ATBASE](\pmidx,\pmidy)
\global\advance \pmidx by -1200
\put(\pmidx,2300){$\nu$}
\end{picture}
\end{center}
\end{document}
```

Note the necessary blank line prior to the \begin{picture} command.

A.6 Exercises for section 4.1.2

```
% Using Phantom Mode.

\begin{picture}(20000,15000)
\drawline\gluon[\E\FLAT](10000,15000)[7]
% Calc lengths of scalars segments and gaps.
\global\Xone=\gluonlengthx
\global\divide\Xone by 8 %  \Xone will be the gap length
\global\Yone=\Xone  % \Xone, \Yone convenient unused variable names.
\double\Yone  %  \Yone will be the segment length
% Draw fermions:
\drawline\fermion[\W\REG](\gluonfrontx,\gluonfronty)[\gluonlengthx]
\drawline\fermion[\S\REG](\gluonfrontx,\gluonfronty)[\gluonlengthx]
\drawline\fermion[\W\REG](\fermionbackx,\fermionbacky)[\gluonlengthx]
% Draw scalars
\global\gaplength=\Xone  \global\seglength=\Yone
\drawline\scalar[\E\REG](\gluonbackx,\gluonbacky)[3]
\global\gaplength=\Xone  \global\seglength=\Yone
\drawline\scalar[\S\REG](\gluonbackx,\gluonbacky)[3]
\global\gaplength=\Xone  \global\seglength=\Yone
\drawline\scalar[\E\REG](\scalarbackx,\scalarbacky)[3]
% Now the photon.  Need to know it's length so use \phantom
\startphantom
\drawline\photon[\E\FLIPPED](0,0)[8]  % Can draw from anywhere
\stopphantom      % Now we have the photon's length!!
\negate\photonlengthx
\global\advance\gluonlengthx by \photonlengthx  % the difference
\global\divide\gluonlengthx by 2
\drawline\fermion[\E\REG](\fermionfrontx,\fermionfronty)[\gluonlengthx]
\drawline\photon[\E\FLIPPED](\pbackx,\pbacky)[8]  % Exactly same as in phantom mode.
\drawline\fermion[\E\REG](\pbackx,\pbacky)[\gluonlengthx]
% This is even easier using stemmed photons!
\end{picture}
```

The difficulty in rotating this diagram through 45 degrees is that we would require very short "stems" on the ends of the photon and there is a minimum length which diagonal lines may be drawn. One could try to connect the ends by a semi-circle (which may be slightly smaller), a kinked pair of vertical and horizontal lines, a large dot at the vertex *etc.*

Almost any diagram may be drawn without recourse to phantom mode given enough effort and ingenuity. In this case one could

```
1)  Draw the Gluon
2)  From the point (\pmidx,\pmidy) go down a distance of
    \gluonlengthx (or \plengthx)
3)  Draw one half of the photon to the left and one half to the right
4)  Draw fermions of length \photonfrontx minus \gluonfrontx or
    \gluonbackx minus \photonbackx at either end
5)  Draw the fermions and scalars as before.
```

A.7 Exercise for section 4.3.2

```
\hskip 1.25in
\begin{picture}(8000,8000)
% Three-gluon vertex:
\stemvertex1\vertexlink3\drawvertex\gluon[\N 3](0,0)[4]
\drawline\fermion[\SW\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\SE\REG](\vertexonex,\vertexoney)[2000]
\drawline\fermion[\N\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\W\REG](\vertextwox,\vertextwoy)[2000]
% Four-gluon vertex:
\global\stemlength=1500
\stemvertex3\vertexlink4\drawvertex\gluon[\NE 4](\vertexthreex,\vertexthreey)[3]
\drawline\fermion[\W\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\N\REG](\vertextwox,\vertextwoy)[2000]
\drawline\fermion[\E\REG](\vertexthreex,\vertexthreey)[2000]
\drawline\fermion[\N\REG](\vertexthreex,\vertexthreey)[2000]
% Add loops to line four of the four-gluon vertex (linked above with \vertexlink4).
\global\stemlength=1500
\backstemmed\drawline\gluon[\SE\REG](\vertexfourx,\vertexfoury)[2]
\drawarrow[\SE\ATTIP](\pbackx,\pbacky)
\end{picture}
```

A.8 Exercises for Section 4.3.3

The capped part of the diagram in section 2.10.2 was produced by:

```
\begin{picture}(25000,10000)
\THICKLINES
\put(24000,7000){\circle{3000}}
% We request a circle of diameter 3000 so LaTeX will use it's maximum:  2800.
% First establish the length of half of the doubly capped gluon (since only the
\startphantom     %                                        back may be capped).
\drawline\gluon[\E\REG](0,0)[2]\gluoncap
\stopphantom
\pbackx=22600 \pbacky=7000   % Establish left edge of circle with diameter=2800.
\global\multiply \plengthx by -1    \global\multiply \plengthy by -1
\global\advance \pbackx by \plengthx  \global\advance \pbacky by \plengthy
\drawline\gluon[\E\REG](\pbackx,\pbacky)[2]\gluoncap  % Draw gluon TO the circle
\drawline\gluon[\W\FLIPPED](\gluonfrontx,\gluonfronty)[2]\gluoncap
\drawline\fermion[\NW\REG](\gluonbackx,\gluonbacky)[2000]
\drawline\fermion[\SW\REG](\gluonbackx,\gluonbacky)[2000]
\gluonbackx=25400 \gluonbacky=7000 % Establish right circle edge: diameter=2800.
\negate\gluonlengthx    \negate\gluonlengthy  % Repeat on the right-hand side.
\global\advance\gluonbackx by \gluonlengthx
\global\advance\gluonbacky by \gluonlengthy
\drawline\gluon[\W\FLIPPED](\gluonbackx,\gluonbacky)[2]\gluoncap
\drawline\gluon[\E\REG](\gluonfrontx,\gluonfronty)[2]\gluoncap
\drawline\fermion[\NE\REG](\gluonbackx,\gluonbacky)[2000]
```

```
\drawline\fermion[\SE\REG](\gluonbackx,\gluonbacky)[2000]
\advance \gluonfrontx by -6800  % Experiment or measure out ``CAPPED'' by ruler.
\put(\gluonfrontx,2000){CAPPED}
\THINLINES
\end{picture}
```

Note that an effective radius of 1400, instead of 1500, was used since the largest circle which LaTeX can draw has a diameter of 28 points.

The concluding exercise of 4.3.3 may be drawn with:

```
\begin{picture}(20000,21000)
% Work our way from the upper left to lower right.
%
\global\stemlength=1500
\backstemmed\drawline\photon[\SE\FLIPPED](0,20000)[8]
\global\advance\pbackx by 1000       % Move to the centre
\global\advance\pbacky by -1000      % of the circle.
\THICKLINES\put(\pbackx,\pbacky){\circle{2830}}\THINLINES   % The circle (thick)
\global\advance\pbackx by 1000       % Move to the SW side
\global\advance\pbacky by -1000      % of the circle.
\global\stemlength=1500
% The extra length on this vertex line is from one loop plus a link.
% Note that the following line is both stemmed and linked.
\frontstemmed\drawline\gluon[\SE\FLIPPED](\pbackx,\pbacky)[1]\gluonlink
\vertexcap2\vertexcap3\drawvertex\gluon[\SE 3](\gluonbackx,\gluonbacky)[3]
% Alternately we could have \vertexlink1\vertexcap2\vertexcap3\drawvertex...
% Now draw the fermions, arrows and labels:
\drawline\fermion[\SW\REG](\vertexthreex,\vertexthreey)[2000] % Southmost gluon.
\drawarrow[\SW\ATBASE](\pmidx,\pmidy)
\global\advance\fermionbackx by -700
\global\advance\fermionbacky by -450
\put(\fermionbackx,\fermionbacky){$q$}
\drawline\fermion[\SE\REG](\vertexthreex,\vertexthreey)[2000] % Eastmost gluon.
\drawarrow[\NW\ATBASE](\pmidx,\pmidy)
\global\advance\fermionbackx by 50
\global\advance\fermionbacky by -450
\put(\fermionbackx,\fermionbacky){$\bar q$}
\drawline\fermion[\NE\REG](\vertextwox,\vertextwoy)[2000]
\drawarrow[\NE\ATBASE](\pmidx,\pmidy)
\global\advance\fermionbackx by 50
\global\advance\fermionbacky by -250
\put(\fermionbackx,\fermionbacky){$q$}
\drawline\fermion[\SE\REG](\vertextwox,\vertextwoy)[2000]
\drawarrow[\NW\ATBASE](\pmidx,\pmidy)
\global\advance\fermionbackx by 50
\global\advance\fermionbacky by -750
\put(\fermionbackx,\fermionbacky){$\bar q$}
\end{picture}
```

A.9 Exercises for section 4.5

A:

This calligraphic masterpiece was created by:

```
\begin{picture}(28000,8000)
%T:
\drawline\gluon[\N\CENTRAL](0,0)[5]
\global\Yone=\gluonbacky
\drawline\fermion[\W\REG](\gluonbackx,\gluonbacky)[4000]
\drawline\fermion[\E\REG](\gluonbackx,\gluonbacky)[4000]
\global\advance\pbackx by 1000
%H:
\drawline\fermion[\N\REG](\pbackx,\gluonfronty)[\gluonlengthy]
\drawline\photon[\E\REG](\pmidx,\pmidy)[5]
\drawline\fermion[\N\REG](\pbackx,\gluonfronty)[\gluonlengthy]
\global\advance\pbackx by 1000
%E:
\drawline\fermion[\N\REG](\pbackx,\gluonfronty)[\gluonlengthy]
\global\advance\pmidy by -300
\drawline\gluon[\E\FLIPPEDCURLY](\pmidx,\pmidy)[4]
\global\advance\plengthx by 500
\drawline\fermion[\E\REG](\gluonfrontx,0)[\plengthx]
\drawline\fermion[\E\REG](\gluonfrontx,\Yone)[\plengthx]
\global\advance\pbackx by 4000
%O:
\drawloop\gluon[\NW 8](\pbackx,300)
\global\advance\loopbackx by 5500
%R:
\drawline\gluon[\S\CURLY](\loopbackx,\Yone)[3]
\global\Xone=\boxlengthy  \double\Xone  \multroothalf\Xone
\put(\pmidx,\pmidy){\oval(\Xone,\boxlengthy)[r]}
\drawline\fermion[\S\REG](\pbackx,\pbacky)[\pbacky]
\global\advance\pfrontx by 500
\global\advance\pfronty by 50
\drawline\photon[\SE\REG](\pfrontx,\pfronty)[4]
%Y:
\global\advance\pbackx by 2600
\drawline\photon[\N\CURLY](\pbackx,0)[3]
\global\Ytwo=\Yone
\negate\plengthy
\global\advance\Ytwo by \plengthy
\double\Ytwo  \multroothalf\Ytwo
\drawline\fermion[\NE\REG](\photonbackx,\photonbacky)[\Ytwo]
\drawline\fermion[\NW\REG](\photonbackx,\photonbacky)[\Ytwo]
\end{picture}
```

B:

The ‘balloon in the tree’ diagram:

```
\begin{picture}(28000,28000)
\THICKLINES
%set up position of bottom of loop
\startphantom
\drawloop\gluon[\NE 0](0,0)
\stopphantom
\global\Xone=\loopfrontx  % diameter of ‘true’ loop
\drawloop\gluon[\S 7](12000,18000)
\global\Xtwo=\gluonbackx
\global\Ytwo=\gluonbacky
% draw gluon vertex
\global\advance\loopmidy by \Xone
\global\stemlength=400   % lengthen the stem since this is drawn in BOLD
\stemvertex1\drawvertex\gluon[\S 3](\loopmidx,\loopmidy)[3]
% Determine diameter and centre of fermion loop
\negate\Xtwo
\global\advance\Xtwo by \loopfrontx
\global\Xthree=\Xtwo %  Store.  Will use shortly
\double\Xtwo  % \Xtwo is now the diameter of the fermion loop
\put(\loopfrontx,\Ytwo){\circle{\Xtwo}}
% Lastly the photon
% This begins located at root 1/2 times radius in x & y direction from centre
\multroothalf\Xthree %  This is why we stored it.
\global\advance\loopfrontx by \Xthree
\global\advance\Ytwo by \Xthree
\drawline\photon[\NE\FLIPPED](\loopfrontx,\Ytwo)[5]
\end{picture}
```

In phantom mode we draw a ‘central loop’ (extent=‘0’). This gives us the position of the *east-most* point of a complete loop and thus the true loop “radius”. By symmetry we now use this to find the south-most position on an *incomplete* loop (extent of seven). We store the (negative of the) “radius” under the name `\Xone` and then draw the gluon loop, clockwise, storing the endpoints as `(\Xtwo,\Ytwo)`. We next draw the three-gluon vertex beginning at the bottom of the gluon loop. To find this point we subtract the radius (\Xone) from the midpoint of the loop, (\loopmidx,\loopmidy).

Next we draw the fermion loop. We accomplish this by finding the radius of the circle, which is the difference between the initial and final x (or y) coordinates of the gluon loop. This is now stored as `\Xthree`. The centre is simply at the x coordinate of the front of the gluon loop and the y coordinate of the rear. Finally we use \multroothalf in order to find the spot, 45° around the circle, from which to draw the photon.

Appendix B

Syntax and Features

This appendix summarizes the features available in **FEYNMAN** and their usage.

B.1 Lines

B.1.1 Arguments and Syntax of the \drawline command

The \drawline command:

```
\drawline |type| [|direction| |style|] (|starting x-co-ord|,|y-co-ord|)[|length|]

         \fermion    \N    \REG              (integer,integer)      extent (cpt)
         \scalar     \NE   \FLIPPED          (variable,integer)     # segments
         \photon     \E    \CURLY            (integer,variable)     # half-wiggles
         \gluon      \SE   \FLIPPEDCURLY     (variable,variable)    # loops
         \especial   \S    \FLAT             ALL IN CENTIPOINTS     # units
                     \SW   \FLIPPEDFLAT      (1000 cpt = 1/3 cm)    (integer/
                     \W    \CENTRAL                                  variable)
                     \NW   \FLIPPEDCENTRAL
                           \LONGPHOTON
                           \FLIPPEDLONG
                           \SQUASHED
```

Where the length corresponds to the particle *type* in the same column. Not all combination of directions and styles are permitted. Fermions, for instance, only appear in the `\REG` style. Integer-valued positions and lengths will be in *centipoints* (cpt) with approximate extent of $\frac{1}{3000}$ *cm* or $\frac{1}{7000}$ *in.* Variables may either be user-defined (see next section) or **FEYNMAN**-defined, such as `\pmidx`, `\gluonlengthy`, `\gaplength`, or `\fermioncount`.

Examples:

```
\drawline\photon[\E\FLIPPEDFLAT](\particlebackx,25000)[9]
\drawline\fermion[\SW\REG](\vertexfourx,\vertexfoury)[\fermionlengthy]
```

B.1.2 Line Modifiers

The following is a list of modifiers which can be used to alter or embellish particle lines. The number in parentheses is the section in which the feature is discussed.

```
\bigphotons (2.9.2)
\documentstyle (1.6)
\gaplength (2.8.2)
\gluoncap (4.3.3)
\gluonlink (4.3.1)
\linethickness (LaTeX manual section C.13.3)
\THICKLINES; \THINLINES; \thicklines; \thinlines (2.4)
\seglength (2.8.2)
\stemmed,\frontstemmed,\backstemmed (4.3.2)
\stemlength (4.3.2)
```

B.2 Vertices

B.2.1 Arguments and Syntax of the \drawvertex command

The \drawvertex command:

```
\drawvertex |type| [|direction| |number|] (|starting x-co-ord|,|y-co-ord|)[|length|]

           \photon     \N          3         (integer,integer)       # half-wiggles
           \gluon      \NE         4         (variable,integer)      # loops
                       \E                    (integer,variable)
                       \SE                   (variable,variable)
                       \S                    ALL IN CENTIPOINTS
                       \SW                   (1000 cpt = 1/3 cm)     (integer/
                       \W                                             variable)
                       \NW
```

Where the length corresponds to the particle *type* in the same column. The vertex is drawn starting from the specified (x,y) co-ordinate with either three or four lines. The first line is drawn in the indicated direction, terminating at the hub (midpoint) of the vertex. It is called *line one*. The remaining lines are drawn radiating from the vertex and numbered counterclockwise as line two, line three (and line four if 'number'=4). Every line will have the same number of gluon loops or photon wiggles. Examples of other vertices, such as a fermion-fermion-photon vertex, are discussed throughout the manual.

Examples:

```
\drawvertex\photon[\E 3](\vertexonex,-5000)[5]
\drawvertex\gluon[\SW 4](\vertexfourx,\vertexfoury)[4]
```

B.2.2 Vertex Modifiers

The following is a list of modifiers which can be used to alter or embellish vertices. The number in parentheses is the section in which the feature is discussed.

```
\bigphotons (2.9.2)
```

```
\documentstyle (1.6)
\flipvertex (3.7)
\THICKLINES; \THINLINES; \thicklines; \thinlines (2.4)
\stemlength (4.3.2)
\stemvertex, \stemvertices (4.3.2)
\vertexcap, \vertexcaps (4.3.3)
\vertexlink, \vertexlinks (4.3.1)
```

B.3 Returned Parameters

The following is a list of all arguments returned by \drawline, \drawvertex and \drawloop. Arguments such as `\photonfronty` will only be returned when a photon is drawn. All returned parameters are *globally* defined and immediately supersede the previous value of the variable. Thus if a gluon were drawn (or a gluon vertex) the values of `\pbackx` and `\gluonlengthx` would be updated but `\photonfrontx` would retain its former value. When a vertex or loop is drawn line parameters, such as \pmidx, will refer to the *last* line or line portion drawn. Parameters commencing with **vertex** are only returned by \drawvertex. Parameters commencing with **loop** are only returned by \drawloop. Note that some of these parameters may be altered by stems, caps, links and arrows (see appropriate sections).

`\boxlengthx,\boxlengthy:`	`The absolute valued (x,y) extent of the line.`
`\fermionbackx,\fermionbacky:`	`The (x,y) co-ordinates of the back of the line.`
`\fermioncount`	`The number of fermions printed thus far.`
`\fermionfrontx,\fermionfronty:`	`The (x,y) co-ordinates of the front of the line.`
`\fermionlength`	`The total length of the fermion line.`
`\fermionlengthx,\fermionlengthy:`	`The (x,y) extent of the line.`
`\gluonbackx,\gluonbacky:`	`The (x,y) co-ordinates of the back of the line.`
`\gluoncount`	`The number of gluons printed thus far.`
`\gluonfrontx,\gluonfronty:`	`The (x,y) co-ordinates of the front of the line.`
`\gluonlengthx,\gluonlengthy:`	`The (x,y) extent of the line.`
`\lastline:`	`The name of the last drawn particle line.`
`\loopbackx,\loopbacky:`	`The (x,y) co-ordinates of the back of the loop.`
`\loopfrontx,\loopfronty:`	`The (x,y) co-ordinates of the front of the loop.`
`\particlebackx,\particlebacky:`	`The (x,y) co-ordinates of the back of the line.`
`\particlefrontx,\particlefronty:`	`The (x,y) co-ords of the front of the line.`
`\particlelengthx,\particlelengthy:`	`The (x,y) extent of the line.`
`\particlemidx,\particlemidy:`	`The (x,y) co-ordinates of the middle of the line.`
`\pbackx,\pbacky:`	`= \particlebackx,\particlebacky`
`\pfrontx,\pfronty:`	`= \particlefrontx,\particlefronty`
`\plengthx,\plengthy:`	`= \particlelengthx,\particlelengthy`
`\pmidx,\pmidy:`	`= \particlemidx,\particlemidy`
`\photoncount`	`The number of photons printed thus far.`
`\photonbackx,\photonbacky:`	`The (x,y) co-ordinates of the back of the line.`
`\photonfrontx,\photonfronty:`	`The (x,y) co-ordinates of the front of the line.`
`\photonlengthx,\photonlengthy:`	`The (x,y) extent of the line.`
`\scalarcount:`	`The total number of scalars printed thus far.`
`\scalarbackx,\scalarbacky:`	`The (x,y) co-ordinates of the back of the scalar.`
`\scalarfrontx,\scalarfronty:`	`The (x,y) co-ordinates of the front of the scalar.`
`\scalarlengthx,\scalarlengthy:`	`The (x,y) extent of the scalar.`

`\unitboxlength,\unitboxheight:`	`The (x,y) extent of one unit of the last line.`
`\unitboxnumber:`	`The 'length' of the previous line in 'units'.`
`\vertexcount:`	`The number of vertices printed thus far.`
`\vertexonex,\vertexoney:`	`The (x,y) co-ordinates of the back of line one.`
`\vertextwox,\vertextwoy:`	`The (x,y) co-ordinates of the back of line two.`
`\vertexthreex,\vertexthreey:`	`The (x,y) co-ordinates of the back of line three.`
`\vertexfourx,\vertexfoury:`	`The (x,y) co-ordinates of the back of line four.`
`\vertexmidx,\vertexmidy:`	`The (x,y) co-ordinates of the middle of the vertex.`

B.4 Arrows and Loops

The other two \draw commands in **FEYNMAN** are \drawarrow and \drawloop.

B.4.1 \drawarrow

The syntax is:

```
\drawarrow[|direction| |configuration|] (|starting x-co-ord|,|y-co-ord|)

              \N          \ATBASE            (integer,integer)
              \NE         \ATTIP             (variable,integer)
              \E                             (integer,variable)
              \SE                            (variable,variable)
              \S                             ALL IN CENTIPOINTS
              \SW                            (1000 cpt = 1/3 cm)
              \W
              \NW
              \LINEDIRECTION
              \LDIR
```

An arrowhead is produced at the specified (x,y) co-ordinates in the specified direction. If \LDIR or \LINEDIRECTION is used then the direction will be that of the most recently drawn line in the current picture. If the previous line was drawn by the \drawvertex command then the *initially specified* vertex direction will be used. If \ATBASE is used then the specified co-ordinates refer to the *base* of the arrow. If \ATTIP is used then the specified co-ordinates refer to the *tip* of the arrow.

The following parameters are returned by \drawarrow:

`\arrowlength`	`The (absolute) length of the arrow drawn (cp).`
`\boxlengthx,\boxlengthy`	`The (x,y) co-ordinatess of the TIP of the arrow.`

Other variables, such as \pbackx,y and \photonlengthx,y are unaffected by \drawarrow. Arrows may be emboldened, via \THICKLINES, independently of the line to which they are attached. Arrows will remain invisible in phantom mode. Examples are:

```
\drawarrow[\LDIR\ATTIP](\pmidx,\pmidy)
\drawarrow[\S\ATBASE](0,0)
```

B.4.2 \drawloop

This was detailed in section 4.4. The syntax is:

```
\drawloop |type| [|direction| |extent|] (|starting x-co-ord|,|y-co-ord|)

          \gluon     \N         0            (integer,integer)
                     \NE        1            (variable,integer)
                     \E         2            (integer,variable)
                     \SE        3            (variable,variable)
                     \S         4            ALL IN CENTIPOINTS
                     \SW        5            (1000 cpt = 1/3 cm)
                     \W         6
                     \NW        7
                                8
```

A *central* loop is drawn if the extent is specified as zero (section 4.4.2). Returned parameters are:

```
\loopbackx,\loopbacky     co-ords of opposite point of loop (non-central loop)
                          co-ords of right-most point of loop (central loop)
\loopfrontx,\loopfronty   co-ords of beginning point of loop (non-central loop)
                          co-ords of left-most point of loop (central loop)
\loopmidx,\loopmidy       co-ords of geometric middle point of loop
\pbackx,\pbacky           co-ords of end point of loop
\gluonbackx,\gluonbacky   co-ords of end point of loop (for a gluon loop)
```

\loopbackx,y and \loopmidx,y are only assigned values if at least half of a loop has been drawn, that is if the loop 'extent' is 0,4,5,6,7 or 8. Example:

```
\drawloop\gluon[\NE 3](\pbackx,\pbacky)
\drawloop\gluon[\E 0](0,0)
```

B.5 Placement and Storage Features

Placement and storage facilities were discussed in sections 4.1 and 4.2. The following is a list followed by the syntax or typical usage. In cases where the feature is principally discussed in other sections these are listed in parentheses. Parameters such as \gluonlengthx and \vertexmidx have been listed in appendix B.3.

```
\addtocounter: \addtocounter{<variable name><increment>}    (See 4.1.1)
\advance:      \global\advance<variable> by <increment(number or variable)>
               Example:   \global\advance \X by \Y
\divide:       \global\divide<variable> by <divisor(number or variable)>
\double:       \double<variable>  Eg:  \double\Yfive.
\drawandsaveline: \drawandsaveline '<line name>' as <arguments as with \drawline>
\drawoldpic:   \drawoldpic<line name>(x,y)  Eg: \drawoldpic\mygluon(0,\pbacky)
```

\hskip:	\hskip<length> (1.6) Used to move pictures around on the page.
\lastline:	Stored name of previously drawn line. Eg: \drawoldpic\lastline(0,0)
\multiply:	\global\multiply<variable> by <multiplier(number or variable)>
	Example: \global\multiply \Xtwo by 7
\multroothalf:	\multroothalf<variable>
\negate:	\negate<variable> eg: \negate\plengthx
\newcount:	\global\newcount<your variable name>
	Example: \global\newcount\pmidycopy \global\pmidycopy=\pmidy
\newcounter:	\newcounter{<variable name>} (See 4.1.1)
\newsavebox:	\global\newsavebox{<name>}. Eg: \global\newsavebox{\Bhabha}.
\put:	\put(<x>,<y>){<object>}. Eg: \put(0,\pmidx){\circle{1000}} (2.11.2).
\savebox:	\savebox{<box name>}(<box width>,<box height>)[<position>]{<object>}
\sbox:	\sbox{<name>}{<object>}. Like \savebox. Eg:\savebox{\Mypic}{}.
\setcounter:	\setcounter{<variable name><variable value>} (See 4.1.1)
\startphantom:	\startphantom...text...\stopphantom
\stopphantom:	\startphantom...text...\stopphantom
\the:	Evaluates a variable (section 4.2.1).
\vskip:	\vskip<length> (1.6) Used to move pictures around on the page.
\Xone:	Unassigned variable pre-defined by FEYNMAN.
\Xtwo:	Unassigned variable pre-defined by FEYNMAN.
\Xthree:	Unassigned variable pre-defined by FEYNMAN.
\Xfour:	Unassigned variable pre-defined by FEYNMAN.
\Xfive:	Unassigned variable pre-defined by FEYNMAN.
\Xsix:	Unassigned variable pre-defined by FEYNMAN.
\Xseven:	Unassigned variable pre-defined by FEYNMAN.
\Xeight:	Unassigned variable pre-defined by FEYNMAN.
\Yone:	Unassigned variable pre-defined by FEYNMAN.
\Ytwo:	Unassigned variable pre-defined by FEYNMAN.
\Ythree:	Unassigned variable pre-defined by FEYNMAN.
\Yfour:	Unassigned variable pre-defined by FEYNMAN.
\Yfive:	Unassigned variable pre-defined by FEYNMAN.
\Ysix:	Unassigned variable pre-defined by FEYNMAN.
\Yseven:	Unassigned variable pre-defined by FEYNMAN.
\Yeight:	Unassigned variable pre-defined by FEYNMAN.
=:	\global<variable>=<value(number or variable)>. Eg: \global\X=2.

Appendix C

Errors

In this appendix we discuss some of the more common errors which you might encounter. When the statement `LATEX <FILENAME>` is executed it will produce a number of files including a `<FILENAME>.LIS` file. Error messages will be recorded in this file (which the user often deletes automatically if running a TeX program via a standard user-defined macro). If submitted interactively they will also appear on the terminal screen. Rule one is to check the error message in the LIS file and locate the purportedly offending code sequence. Check for typos, particularly missing backslashes, \end{...}, omitted blank spaces and incautious use of upper or lower case letters. If you cannot understand the error after using this appendix you might try, firstly, the on-line interactive help facility or, failing that, the ERROR sections in the LaTeX and TeX manuals.

C.1 Option Errors

FEYNMAN issues a number of error statements if an incorrect sequence of parameters is issued to a \drawline, \drawvertex or \drawloop command. The most common of these is

```
*** ERROR IN PARTICLE OPTIONS SELECTION ***
```

This means that you asked for a combination of parameters which does not exist or has not been implemented. Examples are asking for a FERMION of any type other than REG, a PHOTON in the \E direction of type FLIPPEDLONG, a 5-gluon vertex, a four-scalar vertex, a photon loop and so forth. The usual result is that, instead of the desired line(s) a series of tiny "Error" statements will appear in your diagram (see section C.2 for an example). Other messages of this type are:

```
          *** ERROR IN PARTICLE TYPE SELECTION ***
  +++ Try with line type \fermion, \scalar, \photon, \gluon (see manual) +++

          *** ERROR IN PARTICLE DIRECTION SELECTION ***
   +++ Try again with direction N, NE, E, SE etc. or see manual +++
```

In the first instance you specified some type of line of type other than \fermion, \scalar, \photon or \gluon. In the second instance a direction other than `\N, \NE, \E, \SE, \S, \SW, \W` or `\NW` was specified (for instance "\NNW" or "E"). This may have been preceded by "Expecting Number..." (see below). Another message in the same vein is

```
*** ERROR:  PARTICLE OF NEGATIVE OR ZERO LENGTH REQUESTED. ***
***         TAKING ABSOLUTE VALUE.                         ***
```

which is self-explanatory. Just remember that most parameters such as \photonlengthx, \plengthy *etc.* may be positive or negative (see section 2.6).

A warning or error statement will be issued if a line of absurd length has been requested. For gluons, for instance, the statements would be

```
*** WARNING *** Gluon with <specified number of loops> loops requested ***
```

if the number of loops which the user had requested was in excess of 40 and further

```
*** Reducing gluon length to 6 loops (max 85) ***
```

if the user had requested in excess of 85 loops and finally

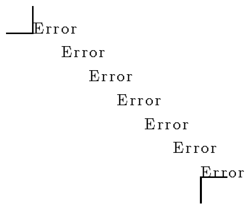
```
*** Probable Cause:  Gluon selected instead of Fermion ***
```

if the user had requested more than 1000 loops.

The latter recognizes a common cause of such problems: copying lines with an editor. When a \drawline\fermion statement for a fermion of length 2000 is copied and altered to generate a photon the user may forget to change the length and thus request a photon of 2000 half-wiggles! The result would be (if not caught) either to run out of TEX capacity or to produce about 30 blank pages on your laser printer.

C.2 Unexpected Results and Other Errors

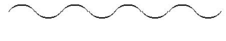
Overcoming exhausted TEX capacity will be dealt with separately in the next section. Here we consider unusual objects which may appear in one's Feynman diagram and a few strange TEX-generated error messages.

The diagram  was produced by

```
\begin{picture}(8000,8000)
\drawline\gluon[\SE\FLAT](0,7000)[6]
\drawline\fermion[\N\REG](\gluonfrontx,\gluonfronty)[1000]
\drawline\fermion[\W\REG](\gluonfrontx,\gluonfronty)[1000]
\drawline\fermion[\S\REG](\gluonbackx,\gluonbacky)[1000]
\drawline\fermion[\E\REG](\gluonbackx,\gluonbacky)[1000]
\end{picture}
```

because gluons in the SE direction cannot appear in the \FLAT mode (see "ERROR IN PARTICLE OPTIONS SELECTION" in C.1).

Unexpected gaps and invisible lines: These may appear as the result of drawing diagonal straight lines of length less than the minimum of 1416 centipoints. This may occur in fermions, scalars (of seglength less than the above), stems and caps. The minimum size of a circle is 401 centipoints. See also section 4.1.2.

Broken horizontal photons, such as  are the result of mis-using (or not using) "\bigphotons" (section 2.9).

Other sorts of errors are:

Expected Number ... You've left a backslash off of a parameter or variable.

```
LaTeX error.  See LaTeX manual for explanation.
              Type  H <return>  for immediate help.
! \begin{picture} ended by \end{document}.
```

or, possibly something like

```
! Missing } inserted.
<inserted text>
                }
<to be read again>
                   \vfil
\newpage ->\par \vfil
                      \penalty -\@M
\clearpage ->\newpage
                      \write \m@ne {}\vbox {}\penalty -\@Mi
\enddocument ->\@checkend {document}\clearpage
                                               \begingroup \if@filesw \immed...

\end #1->\csname end#1\endcsname
                                 \endgroup \@checkend {#1}\if@ignore \global...
l.13 \end{document}
```

You've forgotten to close a grouping with a \end{...}. In this case you forgot the `\end{picture}` statement. If the error is

```
! Use of \picture doesn't match its definition.
```

Then you typed "\picture" instead of "picture".

C.3 TEX Capacity Exceeded

This is the most dreaded of all error messages. It is also the most common.

LATEX allows the user a certain amount of internal working space and then uses up most of it itself. To increase this a TEX wizard must re-compile TEX (generally in Pascal). The most obvious way in which TEX will exceed its limit is if there is too much text. Since **FEYNMAN** uses quite a bit of LATEX's internal resources this limits the size and complexity of diagrams to roughly that of the cover diagram. If gluons are not involved then more complex diagrams are possible. LATEX provides a partial solution to this problem which will be discussed last. When TEX is used the LIS file will end with a summary of memory usage of the form:

```
Here is how much of TeX's memory you used:
 462 strings out of 2520
 4525 string characters out of 29548
 44019 words of memory out of 65501
 2381 multiletter control sequences out of 2500
 19296 words of font info for 73 fonts, out of 35000 for 100
 14 hyphenation exceptions out of 307
 20i,16n,53p,168b,280s stack positions out of 200i,40n,60p,500b,600s
No pages of output.
```

C.3.1 Exhausting Internal Storage

TEX keeps track of a number of internal variables and will cease processing a file should the allocated maximum usage be exceeded. When **FEYNMAN** is being used extensively the most common TEX capacity exceeded messages will be:

Save-Stack: Likely cause is that variables were passed both locally and globally between subroutines too many times. Check that \global precedes all assignment statements, arithmetic operations (except \negate *etc.*) and so forth. In the above LIS file example 280 out of 600 S-stack (save stack) positions were used.

P-Stack: Your macro definitions have exceeded the maximum total number of arguments permitted. This, and the above error, could be caused by a recursive definition gone awry. In the above example 53 out of 60 P-Stack positions were used.

! No room for a new \count.: You have defined too many numerical counters via the \newcount or \newcounter command. Between them TEX, LATEX, and **FEYNMAN**, may use up to 218 out of 234 available counters. Try to re-use old variables, the \Xone...\Yeight available for the user or use some \dimen counters where possible. Typically the last few lines in your LIS file will look like:

```
\OOO=\count234

! No room for a new \count .
\ch@ck ...\else \errmessage {No room for a new #3}
                                                  \fi
\alloc@ ...\advance \count 1#1by\@ne \ch@ck #1#4#2
                                                  \allocationnumber =\count ...
l.36 \newcount\PPP
```

followed by a synopsis of TEX's memory usage.

Multi-letter Control Sequences: Probably too many macro \def's defined. Many such errors will occur if you attempt to input **FEYNMAN** more than once.

C.3.2 Exhausting Memory Size

If your LIS file tells you that you've used 65501 words of memory out of 65501 then you have reached the maximum file 'size' that LATEX will permit. Unless you've made some error, such as having a file \input itself, your choices are to enlarge TEX (this requires a 'wizard', whatever that is), make your program smaller, or break your document up into smaller pieces. The last option would be irritating if, for instance, you were writing a book and had to continually keep track of cross-references between these divisions, page numbering and so forth. LATEX uses the \include facility to alleviate the tedium. The only drawback is that after a break in the document the new position will begin on a fresh page. Individual Feynman diagrams cannot be broken up this way and this limits their ultimate complexity.

To process just a portion of a document place the different portions in a series of files and have a *driving* file which includes the statements

```
\include <file1> \include <file2> \include <file3>... \include <file n>
```

These should appear at the positions in which you want them and may be interspersed with text and commands within the driving file. Prior to TEXing the file the commands `includeonly <file>` should appear for each piece that you want processed on that particular run. For instance, to print out the first part of chapter three of this manual I have a file, which I call FD3A.TEX, containing

```
\documentstyle[11pt]{report}
\input FEYNMAN
\includeonly{FEYNMANDOC3A}
\input FEYNMANDOC.INC
```

where the file FEYNMANDOC.INC, unconditionally input, contains

```
\textheight 700pt \textwidth 450pt
<VARIOUS DEFINITIONS COMMON TO ALL PARTS>

\begin{document}           % End of preamble and beginning of text.

\include{FEYNMANDOCTITLE}
\include{FEYNMANDOCPRELIMS}
\include{FEYNMANDOC1}
\include{FEYNMANDOC2A}
\include{FEYNMANDOC2B}
\include{FEYNMANDOC2C}
                      .
                      .
                      .
\include{FEYNMANDOC4H}
\include{FEYNMANDOC4I}
\include{FEYNMANDOC4J}
\include{FEYNMANDOCEXER}
\include{FEYNMANDOCAPPB}
\include{FEYNMANDOCAPPC}
\end{document}
```

When executed each section will create a number of files such as the `<filename>.AUX` file. These must be retained on your TEX directory as they transmit information about one section to the others. AUX files tell about page numbering, sections *etc.* , BIB files contain the accumulated bibliography, TOC files the table of contents and so forth. For this reason it may take several runs to get your document to come out correctly as each AUX file will be updating itself based upon updates of the other AUX files. TOC files will be the last to come out correctly. To print out this entire document I had a COMMAND file called MAKEDOC.COM which contained:

```
$ SET DEFAULT DRA1:[LEVINE.TEXBOOK]
$ LATEXIT FDTITLE
$ LATEXIT FDPRE
$ LATEXIT FD1
$ LATEXIT FD2A
$ LATEXIT FD2B
$ PURGE FD*.AUX
$ LATEXIT FD2C
       .
       .
       .
$ LATEXIT FD4H
$ LATEXIT FD4I
$ LATEXIT FD4J
$ PURGE FD*.AUX
$ LATEXIT FDADDA
$ LATEXIT FDAPPB
$ LATEXIT FDAPPC
$ PURGE FD*.*
$ PURGE FEYNMANDOC*.*
```

Where LATEXIT is a command which LaTeXs a file, sends it to the printer and then erases the LIS and printer files. The files FD1, FD2A *etc.* are like the small FD3A file above which processed just the first part of chapter three. The PURGE statements remove old versions of the AUX files from your disk. The above file is sent as a batch job by a command file like PRINTDOC which contains

```
$!  Prints FEYNMANDOC.TEX...all of it.
$ SET DEFAULT DRA1:[LEVINE.TEXBOOK]
$ SUBMIT/NOPRINTER/NOTIFY DRA1:[LEVINE.TEXBOOK]MAKEDOC.COM
```

The above is in VAX code but the format on other machines should be apparent.

This work was supported by the UK Science and Engineering Research Council. I would like to thank all of the people at the Cavendish Labs, Cambridge, England, who gave their preferences and comments as to which line styles should be incorporated.

Michael Levine, 1988