Chapter 30 TEXGraph

The graphics package that is a part of INRST$\hat{E}$X adds the following capabilities

- The production of a complete paper/report requiring no additional insertions (except perhaps colour photographs if required).
- The ability to insert a file, for example a plot or halftone, in Encapsulated PostScript Format (EPSF), and to automatically leave enough space in the text.
- The mixing of a vector graphics and TEX text. In fact TEX text may now be printed at any arbitrary angle.

The package requires a sensible PostScript driver. The basic requirements for the driver are as follows:

- The `\special` commands and the other information in the .dvi file are processed in the order they are found in the .dvi file. This is necessary in order to be able to rotate tables and arbitrary text ... among other things.
- The driver does not perform PostScript `save` and `restore` commands around `\special` commands.
- The driver uses the code `ps:` is used to signal a file name and the code `ps::` to signal inline code.
- The driver sets the `currentpoint` before entering a `\special`.

30.1 Command List - TEXGraph

- ⋆ \absoluteposition
- ⋆ \absolutescale
- ⋆ \arcc
- ⋆ \arcn
- ⋆ \arrowheadscale
- ⋆ \arrowheadsize
- ⋆ \arrowheadtype
- ⋆ \avec
- ⋆ \beginsegment
- ⋆ \beginTeXgraph \btg
- ⋆ \cavec
- ⋆ \cav
- ⋆ \centergraph
- ⋆ \centerplotfile
- ⋆ \clvec
- ⋆ \clv
- ⋆ \endsegment
- ⋆ \endTeXgraph \etg
- ⋆ \graphdim
- ⋆ \hgraphsize
- ⋆ \hgraphskip
- ⋆ \hpix
- ⋆ \htext
- ⋆ \includefile
- ⋆ \incscfile
- ⋆ \incscmvfile
- ⋆ \larc
- ⋆ \lcir
- ⋆ \linecap
- ⋆ \linejoin
- ⋆ \lpatt
- ⋆ \lvec
- ⋆ \move
- ⋆ \penwidth
- ⋆ \relativeposition
- ⋆ \relativescale
- ⋆ \rstext
- ⋆ \rtext
- ⋆ \segmentscale
- ⋆ \unitscale
- ⋆ \vgraphsize
- ⋆ \vgraphskip
- ⋆ \vpix
- ⋆ \vtext

30.2 The Inclusion of Encapsulated PostScript

All the commands that are used to include encapsulated PostScript files read the `Boundingbox` and leave enough space in the text for the graphics. The `Boundingbox` may be at either the beginning or the end of the file. However, if the `Boundingbox` is at the the end, it might take TEX some time to read the entire file to find it.

The simplest command for including a graphics file is

$$\backslash\texttt{centerplotfile}\{<\texttt{file name}>\}.$$

For instance the (old) IEEE Communications Society logo was brought in simply by saying

`\medskip`

```
\hrule
\centerplotfile{ieeelogo.ps}
\hrule
\medskip
```
right here.

The `\hrule` commands were put in to show how much space was left by TEXGraph. A slightly more complicated example illustrating both rotation and scaling is given below.

The code for this is

```
\hrule
\centergraph{ \btg
              \incscfile f:{ieeelogo.ps} sc:.3 d:45
              \etg}
\hrule
```

Notice that the computation of the space to leave was incorrect. This is due to the difficulty in performing the computations for obtaining the bounding box for an arbitrarily rotated and scaled figure.

The most general inclusion command is `\incscmvfile` which allows for arbitrary scaling, rotation, and offset. Finally, the rotation, scaling, and placement may be applied around any of the nine text box reference points described in Section 30.4.

30.3 TEXGraph Basics

TEXGraph has a number of capabilities that range from primitive to powerful. The following list of classes is for reference only. More detail will follow with examples in the various sections to follow. This section may be omitted on first reading.

TEXGraph Classes

- Drawing primitives consist of commands for circles, circular arcs, line vectors, vectors with arrowheads, spline curves, spline curves with arrowheads, line dotting commands, and various shades of gray for fills.
- There are commands for changing whether the distances specified are to be interpreted absolutely, with respect to an origin or relatively with respect to the present position.
- There is a command to change the (pen)width of the drawn lines and to change the interpreted scale of the units of distance.
- There is a command to change the default units. However, since arbitrary scaling of units is allowed, this command is of limited use except for those that visualize distances in different dimensions. The dimensions available are those available in INRSTEX.
- There is a "graphical segment" which can be placed at any specific location. Distances inside a graphical segment are interpreted with respect to the segment origin. The distances inside a segment may be either relative or absolute with respect to the segment origin. There is a *segment scale* which allows for segments to be scaled relatively with respect to an overall graph.
- Text, with all the capabilities of INRSTEX may be written at any arbitrary angle.
- TEXGraph keeps track of the maximum extent of a graph. These are available after the graph is terminated in the variables `\hgraphsize` and `\vgraphsize`. `\hgraphskip` and `\vgraphskip` will cause a horizontal or vertical skip of the appropriate graphsize. **Unfortunately, these values may be incorrect for a number of reasons.** These are
  - Text may stick out the edge of the graph.
  - A circle or arc center is outside the main graph area.
  - The visible part of a circle or arc may be too far to the right.
  - A vertical segment may have inadvertently reset the maximum positions.

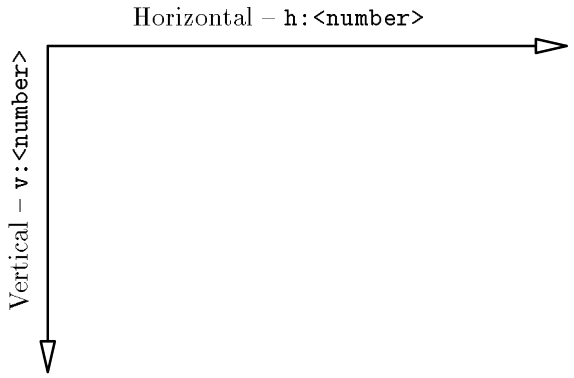
  In some cases it may be necessary to make a fake move to the lower right hand corner of a graph to tell TEXGraph the correct size ... after you have seen a preliminary version and know what it should be.

- All of TEX's definitional capabilities are available. These can be used to replace a complex structure by a single command or to change the surface syntax of TEXGraph.
- Explicit use of grouping in TEXGraph, through {}, or `\begingroup`, `\endgroup` pairs is not allowed. This will foul up the positional record keeping inside. Groups may be produced only through the use of segments, namely the `\beginsegment`, `\endsegment` pair.

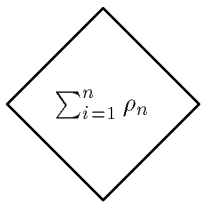
30.4 Simple TEXGraphics

TEXGraph will allow for rather sophisticated graphics. You can use the definition capabilities of INRSTEX to create libraries of symbols for future use.

Directions in TEXGraph are described as horizontal, `h:`, increasing from left to right across the page and vertical, `v:`, increaseing from top to bottom **down** the page. Thus the fundamental directions are



TEXGraph capabilities will be introduced slowly. The following diamond box



was put in place with

```
$$  % a convenient way to use displaymath spacing
\centergraph{
   \beginTeXgraphics % also known as \btg
     \move h:0 v:.5  % moves penup to (0,.5)
     \lvec h:.5 v:0  % draws a line to (.5,0)
     \lvec h:1 v:.5
     \lvec h:.5 v:.1
     \lvec h:0 v:.5
     \textref h:C v:C  % sets the reference point in the center.
     \move h:.5 v:.5
     \htext {$\sum_{i=1}^n \rho_n$} % horizontal text box
    \endTeXgraphics  % also known as \etg
     }
$$
```

The command `\centergraph{<...>}` uses the information about plot size to center the graphics horizontally and to leave enough room vertically.

Arguments for all TEXGraph primitive commands are delimited by spaces. The units of the dimensions are implicit, but changable. The default is inches. Thus `h:<number>` is the distance in the horizontal (to right) and `v:<number>` is the distance in the vertical (down) direction. The locations are absolute distances with respect to an origin in the upper left hand corner.

30.5 Simple Vector Forms

This introduces some baisc drawing commands. There **must** be either a space or an end of line after each `<coordinate>␣`. The dimensional units are implicit. We usually use `<..>` instead of the less economical `<coordinate>`.

⋆ `\move h:<..> v:<..>`

This is a move from the current position to the coordinates given by `h:<..> v:<..>`. The pen is up. It may be used to move the current reference point or to describe a boundary when creating a filled polygon.

⋆ `\lvec h:<..> v:<..>`

This draws a line from the current position to `h:<..> v:<..>`.

⋆ \clvec h1:<..> v1:<..> h2:<..> v2:<..> h3:<..> v3:<..>

This draws a Bezier spline curve from the current point to h3:<..> v3:<..> with respect to the control points h1:<..> v1:<..> and h2:<..> v2:<..>. These curves are very versatile and can create many effects, including loops, by varying the control points.

⋆ \clv (<h1> <v1>) (<h2> <v2>) (<h3> <v3>)

This command is identical to \clvec but with a less verbose syntax.

Lines may be either solid of various widths, or dotted with custom patterns. In addition the ends of the lines may be of different shapes and joined with different defaults. In most cases you will not have to change the defaults in TeXGraph but the capability is there.

⋆ \penwidth <size>

This sets the width of the pen or line in the current units. The default is is .015 (in).

⋆ \linecap <integer>

This sets the end shapes of the lines using PostScript notation. The default is 1 or a round cap. A value of 0 is a butt cap with the line cut off square exactly to its length. A value of 2 is a square cap with the line extended by a square cap the thikness of half its width. For narrow lines the difference is negligible.

⋆ \linejoin <integer>

This sets the form in which lines join (PostScript notation). The default is 1 which corresponds to a round line join. A value of 0 is a mitre or pointed join while a value of 2 is a beveled or flattened join. For thin lines the differences are negligible.

⋆ \lpatt p:<pattern>

This sets the dotting pattern of the line in PostScript notation. The default is a solid line described by {[]}. The simplest form is {[a]} where the line is black for a pixels and white for 1. {[a b]} would mean a pixels black followed by b pixels white. A specification of [a1 b1 a2 b2] would be a cyclic pattern with the line black for the a? pixels and white for the corresponding b?.

TeXGraph alos supplies lines with arrowheads. These may be combined dotting patterns given above but a decision will have to be made as to whether you want the arrowhead itself dotted. There are three styles of arrowheads  triangle T, open O, and filled F. The size may also be varied. There are no restriction on the direction of the arrows. The commands are

⋆ \avec h:<..> v:<..>

This draws a vector with an arrow head from the current position to h:<..> v:<..>.

⋆ \cavec h1:<..> v1:<..> h2:<..> v2:<..> h3:<..> v3:<..>

⋆ \cav (<h1> <v1>) (<h2> <v2>) (<h3> <v3>)

These are the spline versions of the vectors with arrowheads.

⋆ \arrowheadtype t:<type>

The <type> is T O F for triangle, open, or filled respectively.

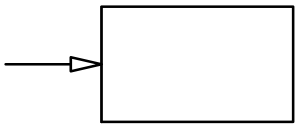
⋆ \arrowheadsize l:<length> w:<width>

The <length> and <width> are in the current units. The default is 0.16in by 0.04in. This size is independent of the current scales. This prevents arrowheads from disappearing as a graph is scaled.

⋆ \arrowheadscale This makes the arrowheadsize sensitive to the segment and unit scales in force when it is called. It is not global but is affected by the grouping imposed by \beginsegment and \endsegment pairs.

For most applications, the \arrowheadsize and the \arrowheadscale will not be needed.

The following example of a (partial) block diagram uses these commands.

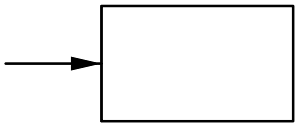


This was produced by the following:

```
\centergraph{\btg
        \move h:0 v:.3
        \avec h:.5 v:.3 % arrow vector leading into box
        \move h:.5 v:.6 % moves to lower lefthand corner of box
        \lvec h:.5 v:0  % four sides
        \lvec h:1.5 v:0
```

```
        \lvec h:1.5 v:.6
        \lvec h:.5 v:.6
          \etg}
```

We can change the type of arrowhead on this and all subsequent arrow vectors by adding adding `\arrowheadtype t:O` before the `\avec ...` to give



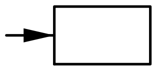
This was produced by

```
\centergraph{\btg
        \move h:0 v:.3
        \arrowheadtype t:F % changes default filled
        \avec h:.5 v:.3 % arrow vector leading into box
        \move h:.5 v:.6 % moves to lower lefthand corner of box
        \lvec h:.5 v:0  % four sides
        \lvec h:1.5 v:0
        \lvec h:1.5 v:.6
        \lvec h:.5 v:.6
          \etg}
```

⋆ `\unitscale <scale>`

This command causes all subsequent units in this or enclosed segments to be scaled by the `<scale>`. Thus `\unitscale 2.54` would have the effect of changing a figure where the units were originally in inches to be in centimetres.

Thus the entire graph is reduced to half size by adding `\unitscale .5␣` at the very beginning.

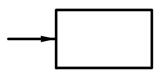


Note that the arrowhead has not changed in size. The commands to do this are

```
\centergraph{\btg
        \unitscale .5  % causes all subsequent units to be multiplied
                       % by .5
        \move h:0 v:.3
        \arrowheadtype t:F % changes default filled
        \avec h:.5 v:.3 % arrow vector leading into box
```

```
        \move h:.5 v:.6 % moves to lower lefthand corner of box
        \lvec h:.5 v:0  % four sides
        \lvec h:1.5 v:0
        \lvec h:1.5 v:.6
        \lvec h:.5 v:.6
          \etg}
```

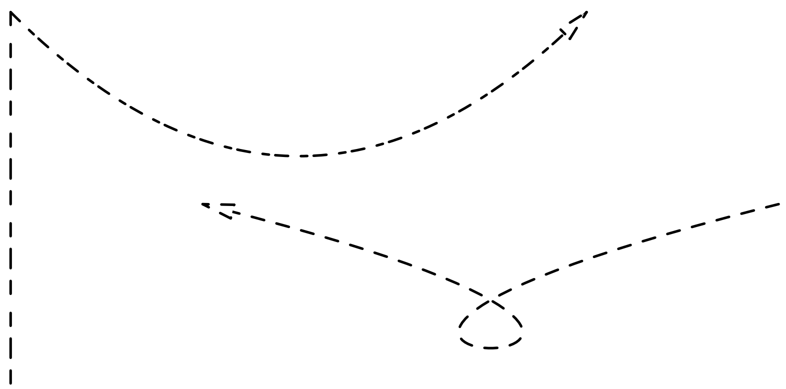
If the arrowhead size needed to be reduced in the same scale it is necessary to add `\arrowheadscale` after the `\unitscale` and before the first arrowvector. Thus we get



This is produced by

```
\centergraph{\btg
        \unitscale .5  % causes all subsequent units to be multiplied
                        % by .5
        \move h:0 v:.3
        \arrowheadscale % makes arrowhead size sensitive to scalings.
        \arrowheadtype t:F % changes default filled
        \avec h:.5 v:.3 % arrow vector leading into box
        \move h:.5 v:.6 % moves to lower lefthand corner of box
        \lvec h:.5 v:0  % four sides
        \lvec h:1.5 v:0
        \lvec h:1.5 v:.6
        \lvec h:.5 v:.6
          \etg}
```

Finally we have an example with dotted Bezier splines.



which was given by

```
\centergraph{ \btg \lpatt p:{[20 30 20]}
                    \lvec h:0 v:2
                    \move h:0 v:0
                    \lpatt p:{[20 20 10 10]}
                   \cav (1 1) (2 1) (3 0)
                    \lpatt p:{[20 20]}
                   \move h:4 v:1
                   \cav (0 2) (5 2) (1 1)
                \etg
            }
```

This is off center because the width computation includes the (5 2) control point of the Bezier curve.

30.5.1 Text in Graphs

All text within TEXGraph is actually typeset by TEX. Since TEX does this so beautifully, it seems counter productive to do it otherwise. The commands for including text at various angles are

⋆ `\htext {<text>}`

This puts the text in a horizontal `\hbox` running from left to right.

⋆ `\vtext {<text>}`

This puts the text in a vertical `\hbox` going upwards.

⋆ `\rtext d:<degrees> t:{<text>}`

This puts text in an `\hbox` with an arbitrary rotation of `<degrees>`.

⋆ `\rstext d:<degrees> sc:<scale> t:{<text>}`

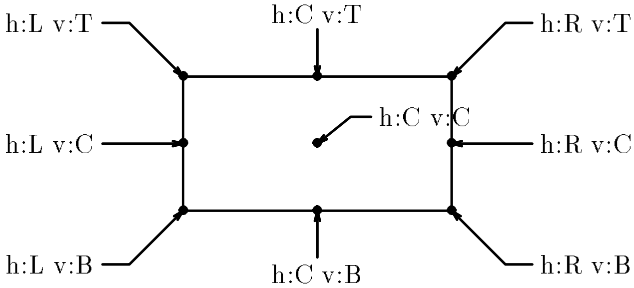
This puts text in an `\hbox` with an arbitrary rotation of `<degrees>` and scale `<scale>`.

Text is placed in an `\hbox`. This box is located at the point in the graph where the `\?text` command is called. In order to make the exact placement of the box in a graph easier, the box is actually oriented at any one of nine reference points. The command is

⋆ `\textref h:<L C R> v:<T C B>`

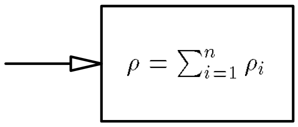
The choices of `L C R` are Left, Center, and Right respectively in the horizontal and vertical positions.

The nine choices are shown below:



These reference points move with the box as it is rotated or scaled.

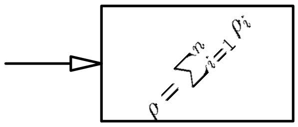
As an example we add the following $\rho = \sum_{i=1}^{n} \rho_i$ to the center of the block diagram used previously to give



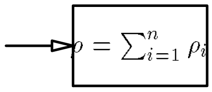
The commands to do this are

```
\centergraph{\btg
        \move h:1.0 v:.3 % moves to center of box
        \textref h:C v:C % sets text reference to dead center
        \htext {$\rho=\sum_{i=1}^{n}\rho_i$} % puts in text
        \move h:0 v:.3
        \avec h:.5 v:.3 % arrow vector leading into box
        \move h:.5 v:.6 % moves to lower lefthand corner of box
        \lvec h:.5 v:0  % four sides
        \lvec h:1.5 v:0
        \lvec h:1.5 v:.6
        \lvec h:.5 v:.6
          \etg}
```

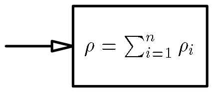
Arbitrary rotation of text or graphics `\rtext`. This example uses `\rtext d:45 t:<text>` in place of the `\htext`.



Note that the text spills out over the edges of the box. There is no guarantee that the text will fit in the box. In fact if we take the horizontal example and add a `\unitscale .7 \arrowheadscale` the following results:

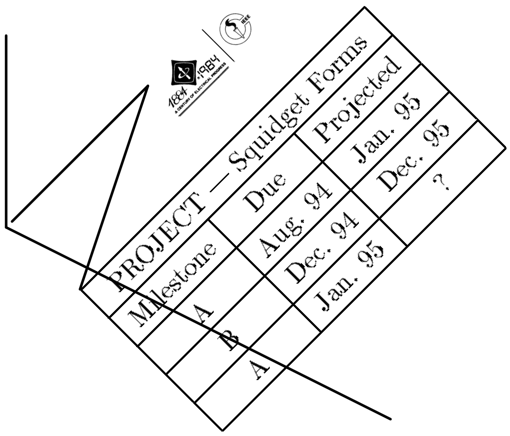


Note that the font is not scaled. If we use `\rstext d:0 sc:.8 t:<...>` we get



30.6 Rotated Graphics and Text

It is possible to place graphics or text at any arbitrary direction on a page. It is only necessary to include it in a `\vtext` for vertical or an `\rtext` for an arbitrary rotation. However, it should be remembered that the reference point is that position on the page where the `\?text` was entered **and** that the text/graphics goes at the appropriate angle. The following shows a simple table angled on the page. To do this correctly, requires that you discover the size of the table.



This was done with the following set of commands. Note that a `\def` was used to make the construction easier to follow. `\lvec` was used rather than `\move` in order to make the origin movements visible.

```
\def\sample{
   \btg
   \lvec h;1 v:0
   \incscalefile f:ieeelogo.ps sc:.25  d:0
   \lvec h:0 v:.5
   \htext{\begintable
            \begintableformat
             &\center
            \endtableformat
            \-
            \br{\:|}\use{3} PROJECT --- Squidget Forms  \er{|}
            \-
            \br{\:|} Milestone | Due      |  Projected    \er{|}
            \-
            \br{\:|}    A      | Aug. 94 | Jan. 95       \er{|}
            \-
            \br{\:|}    B      | Dec. 94 | Dec. 95       \er{|}
            \-
            \br{\:|}    A      | Jan. 95 | ?              \er{|}
            \-
          \endtable}
      \etg}

\medskip
\centergraph{
    \btg \textref h:C v:C
      \lvec h:0 v:1 % sets origin
      \rtext d:45 t:{\sample}
      \lvec h:2 v:2 % sets size of graph ...
    \etg}
\medskip
```

In fact, this example was first done as a separate file to learn the appropriate dimensions and then inputted in the appropriate place.

30.6.1 Circles, Arcs and Fills

TEXGraph uses the circle, arc, and fill commands of PostScript to produce curves. PostScript uses a polygonal fill on the **inside** of the curve. This means that the direction of the arcs are important.

The commands are⋆ `\lcir r:<radius>`

This is a circle with the center at the current loaction and a radius of size `<radius>` in the current dimensions.

⋆ `\arcc r:<radius> sd:<degrees> ed:<degrees>`

This is an “unstroked”arc, ie equivalent to a `\move` in the counter clockwise direction from `sd:<..>` to `ed:<..>`. If this is used in a fill, the center of the circle is the beginning of the path.

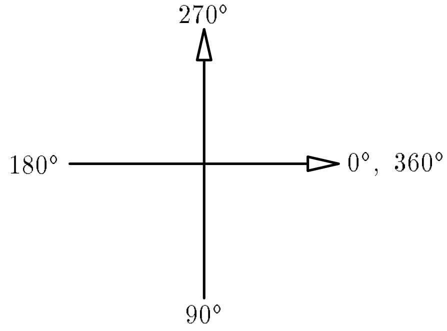
⋆ `\arcn r:<radius> sd:<degrees> ed:<degrees>`

This is the same as `\arcc` except that it is counter clockwise.

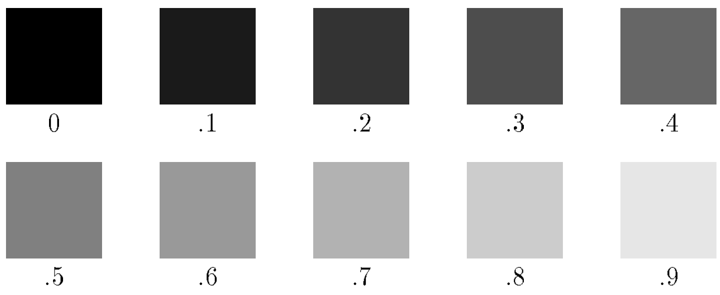
⋆ `\pfill p:<fraction>`

This completes a path a fills the path with a shade of gray with `p:0` being black and `p:1` being white.

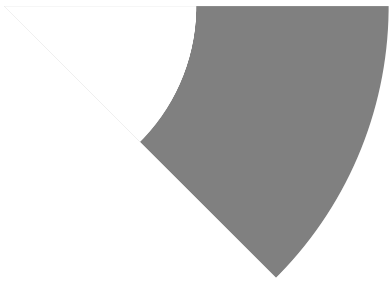
The angles are in the directions shown here.



Some shades of gray are as follows:



An example of a use of arcs and fill is given below.



This logo is given by

```
\centergraph{\btg
               \arcc r:2 sd:0 ed:45
               \pfill p:.5
               \arcc r:1 sd:0 ed:45
               \pfill p:1  % fills the inside with white
               \move h:2 v:1.5 % to leave enough space
              \etg}
```

This shows that a fill of "white" (1) is opaque and obliterates the black or gray underneath. This strategy is necessary in PostScript filling.

30.7 Advanced TEXGraphics and Segments

The most important advanced feature of TEXGraph is the graphical segment. These are delimited by `\beginsegment` `\endsegment` pairs. They are used to create graphical items that can be placed at any specific location in a graph. This capability, combined with TEX's ability to create definitions allows for the creation of libraries of symbols. **Segments are the only groups that should be used within a `\btg` `\etg` pair.**[!] A simple example of the use of this facility was the definition of a `\fillbox` to create the box with the fill pattern shown in Section 30.6.1. The definition was

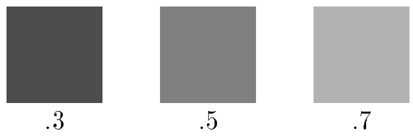
```
\def\fillbox #1 {\move h:.4 v:0
                   \beginsegment
                    \absoluteposition
                    \lvec h:.5 v:0
                    \lvec h:.5 v:.5
```

[!] This is almost true. Groups may be used freely within the argument of either an `\??text`.

```
  \lvec h:0 v:.5
  \pfill p:{#1}
  \move h:.25 v:.55 \textref h:C v:T \htext{#1}
\endsegment
\move h:.4 v:0 % for symmetry
}
```

The relative position in outside the segment allows the boxes to be placed beside each other with even spacing and no additional instructions.

Each segment has a *segment reference point.* This is the location in the enclosing segment or `\btg \etg`, which is really just a special segment. **Absolute positions** within a segment are actually relative to the *segment reference point.* Relative positions are relative. The three boxes below



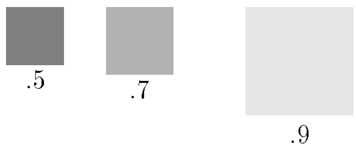
were produced by

```
\medskip
 \centergraph{\btg \relativeposition
               \fillbox .3
               \fillbox .5
               \fillbox .7
              \etg}
\medskip
```

Note that the space after the `#1` in the definition for `\fillbox` means that the numbers are delimited by implicit space at the end of the line. If this space had been left out, only the first digit in each of the pattern numbers would have been read and the other would have been printed at some odd place on the page.

30.7.1 Segment Scaling

There are several scaling parameters that can be used with segments. In addition to the `\unitscale` there is a `\segmentscale`. Roughly speaking the `\segmentscale` is used to change the scale of the present and all enclosed segments while the `\unitscale` is used to set the nominal size of segment object. The major difference between the two is that segment scales accumulate while unitscales do not. An example is obtained using the `\fillbox` in a segment. The following three boxes illustrate the effect.



This example should be studied carefully. The actual commands that produced it are

```
\centergraph{\btg \unitscale .3 % changes units
                \segmentscale 2
                            % doubles scale of all enclosed segments
                            % actual scale is .3*2=.6
                \fillbox .5
                \move h:.8 v:0  % moves to right beyond box
                \beginsegment
                 \unitscale .5 % now unit scale is .5
                 \segmentscale .7
                            % reduces scale by .5 -- actual scale is
                            % .5*2*.7=.7
                 \fillbox .7
                 \move h:.8 v:0
                 \beginsegment
                  \unitscale 1
                  \segmentscale .8  % actual scale is 1*2*.7*.8=1.12
                  \fillbox .9
                 \endsegment
                   % actual scale is back to .5*2*.7=.7
               \endsegment
                   % actual scale is back to .3*2=.6
              \etg}
```

Note how the segment scale accumulates while the unitscale does not. Note also that the scale reverts to that of the enclosing segment after an `\endsegment`.

It is possible to prevent a segment from being affected by the external segment scales by inserting the command `\absolutescale`. This is used for symbols that you wish to have the same size no matter the actual scale of the graph.

30.7.2 Segment Command Summaries

The following is a list of some of the important points concerning segments:

⋆ `\beginsegment`

This begins a graphical segment and a group. All locations within a segment are relative to the origin of the segment.

⋆ `\endsegment`

This terminates a segment and the group. The current location upon leaving a segment is the segment origin.

⋆ `\unitscale <scale>`

The `<scale>` multiplies the present segment and all enclosed segments unless they have their own `\unitscale`. A `\unitscale` may be changed at any time within a segment.

⋆ `\segmentscale <scale>`

The `<scale>` affects the present segment and all enclosed segments unless that segment is protected by an `\absolutescale`. Unlike `\unitscale`, the `\segmentscale` accumulates by multiplying the current scale value. The actual scale used for interpreting position numbers is the product of all of the enclosing segmentscales times the presently active unitscale.

⋆ `\absoluteposition`

All coordinates are to be interpreted with respect to the current origin. This will either be the graph origin or the origin of the current segment.

⋆ `\relativeposition`

Coordinate positions are interpreted relative to the previous position. If the previous command was the drawing of an arc or circle, the previous position is the center of the circle.

30.8 Slide Making

A specialized set of commands along with appropriate templates exist to create a transparency. All slides are "horizontal". `\fullform` slides take up an entire 8.5 by 11 in. page and are intended for display. `\textform` slides are reduced for text. A `\comment <text> //` command is given that is suppressed when the `\fullform` is used but is printed when `\textform` is used. The text in the slides uses a general scaled font family in the file `spfont.tex`. This family is basically a scaled set of `\tenpoint` fonts. An example is given below:

There are two basic forms, the INRS form and a General form. An example of the `\inrsform` is given by

INRS-Télécommunications

HOW TO MAKE SLIDES WITH INRSTÊX

Preliminaries

- **Input the macro package with** `\input inrssl.tex`**.**
- ∘ **Decide whether** `\fullform` **or** `\textform`**.**
- ∗ **Decide whether** `\inrsform` **or** `\genform`**.**

Making it.

- **Use** `\hslide <title>// <body> //`**.**
- **Use "//" as the delimiter.**

```
\input inrssl % brings in the slide macros
\textform % sets textform and size
\formscale .7 % further reduces the size of slide by .7
\beginmidinsert
\inrsform % sets the inrs form
\hslide HOW TO MAKE SLIDES WITH \intex//
\hd{Preliminaries}
\bl
```

```
\lb Input the macro package with {\tt \1input inrssl.tex}.
\lc Decide whether {\tt \1fullform} or {\tt \1textform}.
\last Decide whether {\tt \1inrsform} or {\tt \1genform}.
\el
\hd{Making it.}
\bl
\lb Use {\tt \1hslide <title>// <body> //}.
\lb Use {``//''} as the delimiter.
\el
//
\endmidinsert
```

General Engineering Inc.

This is the title

Split here

This is an internal header

- **A listitem starting with a bullet.**
- ◦ **A listitem starting with a circle.**
- ∗ **A listitem starting with an asterisk.**

An Important Presentation – Jan. 1, 2000

It is possible to add a comment to the bottom of the slide which only shows up when the slide is printed in textform. A comment is terminated with a //

This second slide was given by

```
\topdata = {General Engineering Inc.}
\botdata = {An Important Presentation}
```

```
\slidedate = {Jan. 1, 2000}
\genform % the general form that uses these token strings

\hslide This is the title \cr Split here //
\hd{This is an internal header}
\bl
\lb A listitem  starting with a bullet.
\lc A listitem starting with a circle.
\last A listitem starting with an asterisk.
\el
//  % the // terminates a slide

\comment
It is possible to add a comment to the bottom of the slide which
only shows up when the slide is printed in textform. A comment is
terminated with a {//} //
```

Arbitrary graphics may be included in a slide using normal TEXGraph commands. However, the included graphics may not scale correctly when moving from `\fullform` to `\textform`. Under these conditions is is necessary to include the test `\iftextform <....> \else <...> \fi` to switch between the modes.

The basic commands in order to use the slide package are as follows:

⋆ `\input inrssl`

This command will bring in the INRSTEX slide macro package.

⋆ `\fullform`

This chooses the full size form intended for creating the transparency. Any `\comment` is suppressed.

⋆ `\textform`

This chooses the text size of the slide and does not suppress the `\comment` text.

⋆ `\formscale <scale>`

The entire slide, text, internal graphics, and border, is scaled by the `<scale>` factor. This is done by directly by PostScript.

- `\hslide <title> // <text> //` This is the basic command for making a slide. The title is terminated with a // and the slide body is also terminated with a //.

- `\comment <text> //` This is a comment text which is printed in `\textform` but suppressed in `\fullform`.

- `\inrsform` This will give the standard INRS-Télécommunications format with its logo on the top and nothing on the bottom. Text on the bottom may be added using `\botdata` and `\slidedate`.

- `\genform`

  This is a completely general form where only the border is set.

- `\topdata ={<text>}` This is printed in the `\borderfont` which is the same as the current slide body font. It is printed on the top line above the border with a current position at the left hand corner.

- `\botdata = {<b-text>}`

- `\slidedate = {<s-date>}`

  These two token strings are printed on the bottom line centered and separated by a –, ie as `<b-text> - <s-date>`.

- `\iftextform <...> \else <...> \fi`

  This is a switch that allows for different actions between fullform and textform. `\textform` is .578=1/1.728 the size of `\fullform`.

- `\lb \lc \last`

  These are three short forms for use in listitems starting with •, ∘, and * respectively.

- `\hd{<text>}`

  This gives a "header" inside a slide. It is equivalent to a `\dssshead` in normal text.

30.9 Some Additional Command Forms – TEXGraph

⋆ `\begintexgraph \btg`

This begins TEXGraph. It will bring in the TEXGraph macros the first time it is called. It starts a group and a `\vbox`.

⋆ `\endtexgraph \etg`

Finishes TEXGraph. It terminates the group started by `\btg`.

⋆ `\graphdim <units>`

This sets the default units of TEXGraph. Any valid TEX `<unit>` is valid such as `in, cm, pt,` or `bp`. The current default is `in`.

⋆ `\hgraphsize`

This maximum horizontal excursion of the graph as computed by TEXGraph. It is a dimension.

⋆ `\hgraphskip`

This actually `\hskip \hgraphsize`.

⋆ `\hpix <number>/<unit>`

This is the horizontal resolution in pixels/unit. The default is `300/in`.

⋆ `\vgraphsize`

This maximum vertical excursion of the graph as computed by TEXGraph. It is a dimension.

⋆ `\vgraphskip`

This actually `\vskip \vgraphsize`.

⋆ `\vpix <number>/<unit>`

This is the vertical resolution in pixels/unit. The default is `300/in`.