The mechanism described here is still experimental. Although it could be implemented in MkIV it is only available in LMTX. Already early in MkII we had a positioning mechanism available. At that time we had dvi output and specials were used to track positions. A script calculated the positions that were then fed back into the second run. Later Taco wrote dvipos which sped up the analysis. When pdfTEX evolved, an equivalent positioning mechanism was added there. In MkIV positioning support was optimized as part of the upgrading process a bit but the principles remained the same. The mechanism discussed here works differently and is more local which has pros and cons: it saves a second pass but it only works in a boxed ConTEXt (or course one can unbox a vertical box in the main stream). Performance wise there is probably not much difference (if one discards the initial extra run).

There is not that much to tell about it and examples demonstrate best what can be done with it.

```
\framed [synchronize=yes,align=normal] {%
  test test test 1      \sync{one}{1}test test 2       \crlf
  test test 1           \sync{one}{1}test test test 2 \crlf
  test 1                \sync{one}{1}test 2           \crlf
  test test test test 1 \sync{one}{1}test test 2
}
```

Here we see four anchors being used. The one is just a namespace and the number is a column number.

test test test 1	test test 2
test test 1	test test test 2
test 1	test 2
test test test test 1	test test 2

Here we have two columns:

```
\framed [synchronize=yes,align=normal] {%
    one one     \sync{a}{1}two      \sync{a}{2}three \crlf
    one one     \sync{a}{1}two two \sync{a}{2}three \crlf
    one         \sync{a}{1}two      \sync{a}{2}three \crlf
    one one one \sync{a}{1}two      \sync{a}{2}three
}
```

There are not really columns in the tabular sense but the anchor points perform the same duty as cell separators in a table.

one one	two	three
one one	two two	three
one	two	three
one one one	two	three

The \sync command gets a column number but will automatically increment a row number. The more low level variant is:

```
\framed [synchronize=yes,align=normal] {%
    \syncanchor{one}{1}{1}A
    \syncanchor{one}{2}{1}BBB
    \syncanchor{one}{3}{1}C \crlf
    \syncanchor{one}{1}{2}AA
    \syncanchor{one}{3}{2}C \crlf
    \syncanchor{one}{1}{3}AAA
    \syncanchor{one}{2}{3}BB
    \syncanchor{one}{3}{3}C
}
```

Here a cell is missing which is why we are more explicit with the coordinates.

```
A     BBB C
AA        C
AAA BB    C
```

Although this mechanism is nor really meant for tables you can influence the alignment a bit:

```
\framed [synchronize=yes,align=normal] {%
    \alignanchor{one}{1}{1}{l}A
    \alignanchor{one}{1}{2}{m}AA
    \alignanchor{one}{1}{3}{r}AAA
    \syncanchor {one}{1}{4}\crlf
    \alignanchor{one}{2}{1}{right}B
    \alignanchor{one}{2}{2}{middle}BB
    \alignanchor{one}{2}{3}{left}BBB
    \syncanchor {one}{2}{4}
}
```

Because this is rather verbose it probably makes most sense to use these helpers in macros which is why we have no high level interface (yet).

```
             A            AA        AAA
B BB BBB
```

Of course a valid question is, how does that work with MetaPost graphics so let's give an example of using anchors there. We start with defining a gew useable graphics:

```
\startuseMPgraphic{whatever-1}
    fill OverlayBox withcolor "lightgray" ;
    drawdot anchorxy("one", 0, 1)
        withpen pencircle scaled 1mm
        withcolor "red" ;
    drawdot anchorxy("one", 0, 2)
        withpen pencircle scaled 1mm
```

```
            withcolor "red" ;
        drawdot anchorxy("two", 0, 1)
            withpen pencircle scaled 1mm
            withcolor "red" ;
        drawdot anchorxy("two", 0, 2)
            withpen pencircle scaled 1mm
            withcolor "red" ;
        drawdot anchorxy("three", 0, 1)
            withpen pencircle scaled 1mm
            withcolor "red" ;
        setbounds currentpicture to OverlayBox ;
    \stopuseMPgraphic

    \startuseMPgraphic{whatever-2}
        pair a ; a := anchorxy("one", 0, 1) ;
        pair b ; b := anchorxy("one", 0, 2) ;
        drawarrow
            anchorxy("one", 0, 1) { right } ..
            anchorxy("one", 0, 2)
            withcolor "blue" withtransparency (1,.75) ;
        drawarrow
            anchorxy("two", 0, 1) { down } ..
            anchorxy("two", 0, 2)
            withcolor "blue" withtransparency (1,.75) ;
        drawarrow
            anchorxy("two", 0, 1) { dir 30 } ..
            anchorxy("three", 0, 1)
            withcolor "blue" withtransparency (1,.75) ;
        setbounds currentpicture to OverlayBox ;
    \stopuseMPgraphic
```

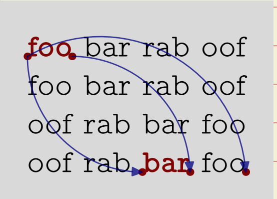
We also define two overlays using them:

```
    \defineoverlay[whatever-1][\useMPgraphic{whatever-1}]
    \defineoverlay[whatever-2][\useMPgraphic{whatever-2}]
```

Because we want the arrow on top, we explicitly flush the foreground (text) between these graphics:

```
    \framed
      [synchronize=yes,
       align=normal,
       offset=10pt,
       frame=off,
       background={whatever-1,foreground,whatever-2}]
      {%
        \markanchor{two}{0}{1}{\red\bf foo}%
        \markanchor{one}{0}{1} bar rab oof\crlf
    foo bar rab oof\crlf
    oof rab bar foo\crlf
    oof rab \markanchor{two}{0}{2}{\red\bf bar}%
    \markanchor{one}{0}{2} foo%
    \markanchor{three}{0}{1}%
  }
```

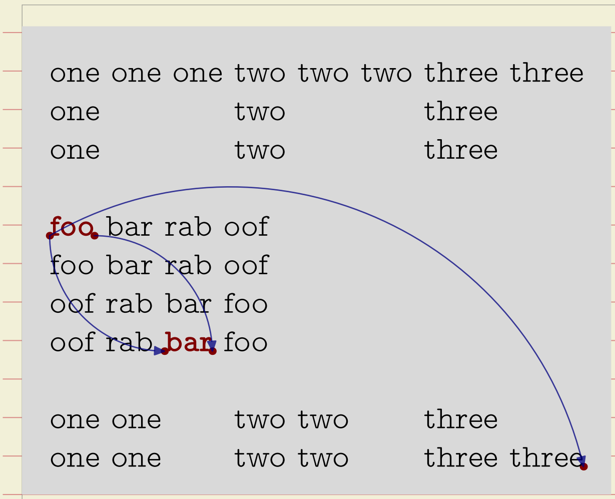
This definition is a bit unreadable because it has to fit on the page, but here is what we get:


foo bar rab oof
foo bar rab oof
oof rab bar foo
oof rab bar foo

All the above trickery can be combined:

```
\framed
  [synchronize=yes,
   align=normal,
   offset=10pt,
   frame=off,
   background={whatever-1,foreground,whatever-2}]
  {%
    one one one \syncanchor{a}{1}{1}two two two
                  \syncanchor{a}{2}{1}three three\par
    one           \syncanchor{a}{1}{2}two
                  \syncanchor{a}{2}{2}three\par
    one           \syncanchor{a}{1}{3}two
                  \syncanchor{a}{2}{3}three\par
    \blank
    \markanchor{two}{0}{1}{\red\bf foo}%
    \markanchor{one}{0}{1} bar rab oof\crlf
    foo bar rab oof\crlf
    oof rab bar foo\crlf
    oof rab \markanchor{two}{0}{2}{\red\bf bar}%
    \markanchor{one}{0}{2} foo%
    \markanchor{three}{0}{1}%
    \blank
    one one \syncanchor{a}{1}{4}two two
              \syncanchor{a}{2}{4}three\par
    one one \syncanchor{a}{1}{5}two two
              \syncanchor{a}{2}{5}three three%
    \markanchor{three}{0}{1}%
  }
```

This again not so readable example gives:



```
one one one two two two three three
one         two         three
one         two         three

foo bar rab oof
foo bar rab oof
oof rab bar foo
oof rab bar foo

one one     two two     three
one one     two two     three three
```

The namespace suggests that we can have more than one alignment set:

```
\framed[synchronize=yes,align=normal]{%
    one one \sync{a}{1}two     \sync{a}{2}three\par
    one one \sync{a}{1}two two \sync{a}{2}three\par
    \blank
    tst tst \sync{b}{1}tst     \sync{b}{2}tst\par
    tst     \sync{b}{1}tst tst \sync{b}{2}tst\par
    \blank
    one     \sync{a}{1}two     \sync{a}{2}three
}
```

This again not so readable example gives the following. Watch how we mix the lines:

```
one one two      three
one one two two three

tst tst tst     tst
tst     tst tst tst

one     two      three
```

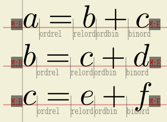
Of course we also want to see how this can be used in math. Take these lines:

```
$a = b + c $\par
$b = c + d $\par
$c = e + f $\par
```

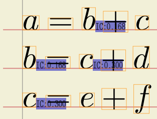
This renders as follows:

$$a = b + c$$
$$b = c + d$$
$$c = e + f$$

If you watch carefully, you will notice that the identifiers and operators don't align vertically. The spacing related to the configured intermathobject spacing parameters:


M:B $a = b + c$ M:E
ordrel relord ordbin binord
M:B $b = c + d$ M:E
ordrel relord ordbin binord
M:B $c = e + f$ M:E
ordrel relord ordbin binord

But here also the italic correction can kick in. And of course the widths of the glyphs differ.



In the next examples we will align the operators:

```
\startsynchronizing
    $a \syncanchor{one}{1}{1}= b \syncanchor{one}{2}{1}+ c$\par
    $b \syncanchor{one}{1}{2}= c \syncanchor{one}{2}{2}+ d$\par
    $c \syncanchor{one}{1}{3}= e \syncanchor{one}{2}{3}+ f$\par
\stopsynchronizing
```

This verbose version gives:

$$a = b + c$$
$$b = c + d$$
$$c = e + f$$

We can use less code:

```
\startsynchronizing
    $a \sync{one}{1}= b \sync{one}{2}+ c$\par
    $b \sync{one}{1}= c \sync{one}{2}+ d$\par
    $c \sync{one}{1}= e \sync{one}{2}+ f$\par
\stopsynchronizing
```

and get:

$$a = b + c$$
$$b = c + d$$
$$c = e + f$$

But it's still not okay, because math spacing is kind of special:

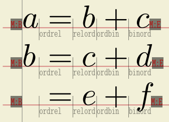
```
\startsynchronizing
    $a \mathsync{1}= b \mathsync{2}+ c$\par
    $b \mathsync{1}= c \mathsync{2}+ d$\par
    $  \mordsync{1}= e \mathsync{2}+ f$\par
```

```
\stopsynchronizing
```

You can best run these examples in a test file and watch the nuances full screen.

$$a = b + c$$
$$b = c + d$$
$$ = e + f$$

With traced math kerning:



With traced math alignment correction: