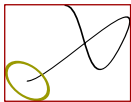



465.69794

When you draw paths, texts or pictures, they are added to the **current picture**. You can manipulate this **current picture**

- current picture can be transformed using any transform. p3 has been slanted .5 in the below picture using.  
`draw p3; currentpicture := currentpicture slanted .5;`  
`pushcurrentpicture; %some other draw commands; popcurrentpicture;`  
 when popped back, the picture will be overlaid on anything that had drawn after push.



Can we place Metapost graphics inline with Text.

Here is a  circle;

## 1 Some Lua Tests

2.7

$$\sqrt{2} = 1.4142135623731$$

## 2 Boxes library for Metapost

The main idea of 'boxes.mp' package for metapost is that one should declare a box as 'boxit.<suffix>(<picture expression>)' This creates pair variables <suffix>.c,<suffix>.n, <suffix>.e>, ... These generated pair variables can then be used for positioning the picture before draw it with a separate command, such as 'drawboxed(<suffix list>)' The argument to 'drawboxed' should be a comma separated list of box names, where box name is a <suffix> with which 'boxit' has been called.

for the command 'boxit.bb(pic)' the box name = 'bb' and contents of box='pic'. 'bb.c' is the position where the centre of 'pic' is to be placed. 'bb.sw, bb.se, bb.ne, and bb.nw' are corners (south west, south east, north west, and north east) of the rectangular path that will surround the 'pic'. variables 'bb.dx' and 'bb.dy' give the spacing between the

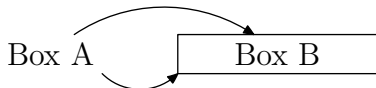


When a boxit macro is called with box name  $b$ , it gives linear equations that force  $b.sw$ ,  $b.se$ ,  $b.ne$ , and  $b.nw$  to be the corners of the rectangle aligned on the  $x$  and  $y$  axis with the box contents centered inside. The values of  $b.c$ ,  $b.dx$  and  $b.dy$  are left unspecified, so that the user could give equations for positioning the boxes. If no such values are given by the user, default values are applied. The default values of  $dx$  and  $dy$  are controlled by *defaultx* and *defaulty*.

If  $b$  represents a boxname, **drawboxed( $b$ )** draws the rectangular boundary of box  $b$ . The bounding rectangle can be accessed as **bpath  $b$** . This is useful for cutting paths that enter the box. For example, if  $a$  and  $b$  are box names and  $p$  is a path from  $a.c$  to  $b.c$ , then

```
drawarrow p cutbefore bpath a cutafter bpath b
```

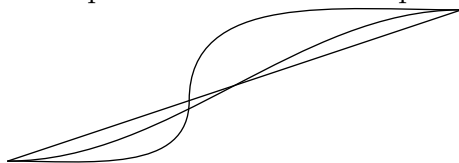
will draw an arrow from edge of box  $a$  to the edge of box  $b$ .



**boxjoin(<equation>)** macro can be used to control the relationship between two boxes. For example, let  $a$  and  $b$  be two boxes then **boxjoin( $a.se=b.sw$ ;  $a.ne=b.nw$ )** will cause the boxes to have same height and line up horizontally.

## 3 Dealing with Unknown points

Meta post can solve linear equations for unknown points (Pairs of numbers).



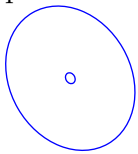


#### 4 Picture Variables

Metapost primitive **addto** is used to add components to a <picture>. Higher level Plan macros like **draw** write to the current picture. Current picture can be saved (push), then set to null, let a higher level macro write to it, and save it to another picture and restored the original picture by popping.

Another newer macro **image** is introduced in Metapost version 0.6. It takes as input a sequence of arbitrary drawing operations and returns a <picture>, without affecting the current picture.

pictures drawn using **image** can be used in other **image** definition to build larger pictures.



#### 5 Shaders

At this point, it is not clear to me whether shaders are supported by metapost or they are being brought in using Context.



## 6 Outlines

Testing

## 7 Including SVGs