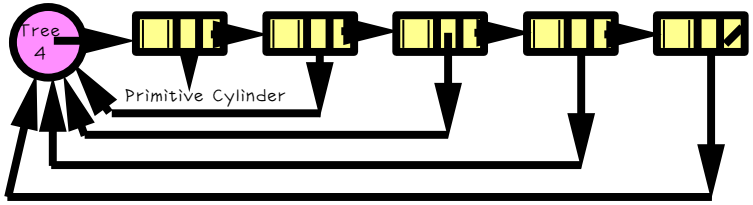
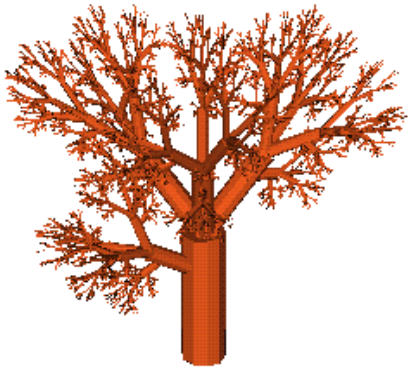
Computer Graphics



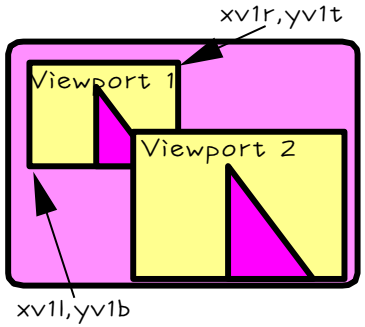


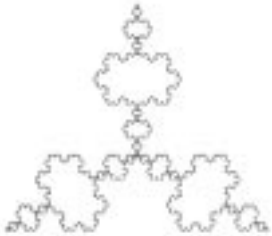
Using Geometric Transformations

by

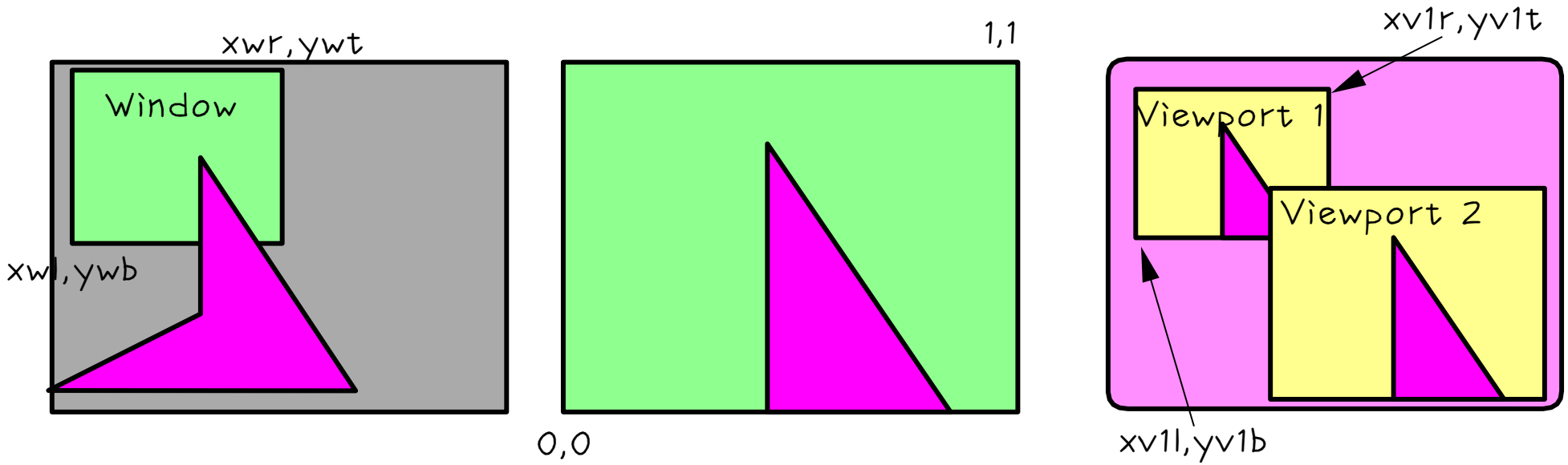
Brian Wyvill

University of Victoria





Windows and Viewports



World Coordinate System

Normalised Device Coordinate System NDC

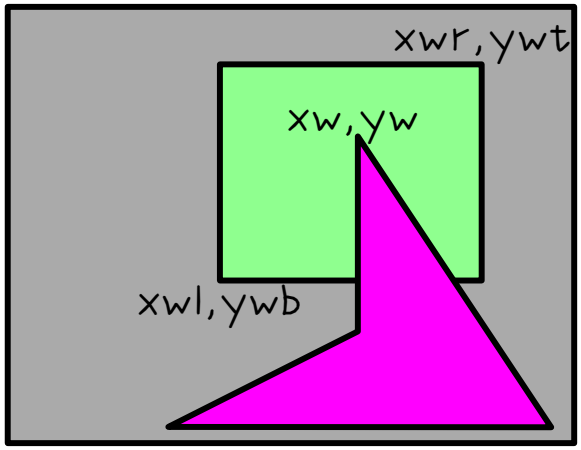
Screen or device Coordinate System

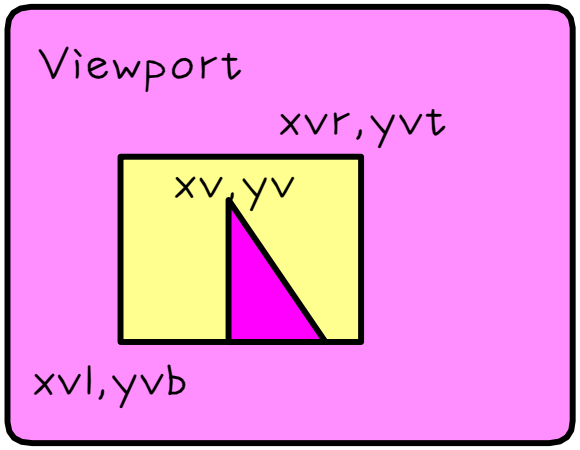
Windows in computer graphics refers to a rectangular area in the users own coordinate system known as the WORLD system. NOT window manager windows (invented much later and misnamed).

Viewport in computer graphics is a window manager window.

Normalised Device Coordinates are useful when displaying on many different devices.

Window to Viewport mapping





Simple mapping keeping scaling in x and y the same

ww = xwr - xwl wh = ywt - ywb arbitrary point in world: xw, yw

vw = xvr - xvl vh = yvt - yvb arbitrary point in viewport: xv, yv

scale = (ww/vw > wh/vh) ? wh/vh : ww/vw

xv = (xw - xwl)*scale + xvl

yv = (yw - ywb)*scale + yvb

$$M = \begin{vmatrix} 1 & 1 & xvl \\ 0 & 1 & yvb \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} scale & 1 & 0 \\ 0 & scale & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} 1 & 1 & -xwl \\ 0 & 1 & -ywb \\ 0 & 0 & 1 \end{vmatrix}$$

$$\begin{vmatrix} xv \\ yv \\ 1 \end{vmatrix} = M * \begin{vmatrix} xw \\ yw \\ 1 \end{vmatrix}$$

Hierarchical Graphics System

In a graphics system primitive models (2D) are composed of a series of straight line segments. Hierarchical models have a head node PICTURE, which points to a linked list of TRANSFORM objects.

PICTURE

1. Pointer to a linked list of TRANSFORM objects. (firstTrans).
2. Integer recursion LIMIT.

TRANSFORM

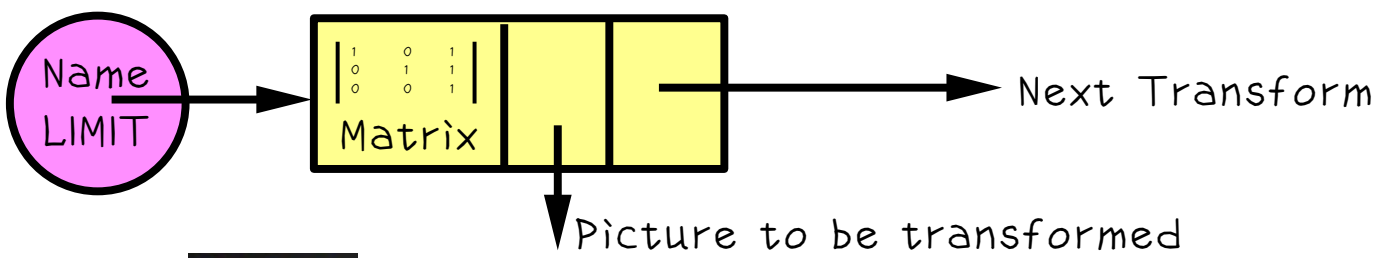
1. A 3 by 3 matrix composed from Translation, Rotation and Scaling operations. MATRIX M

2. A pointer to a PICTURE object or to a PRIMITIVE.

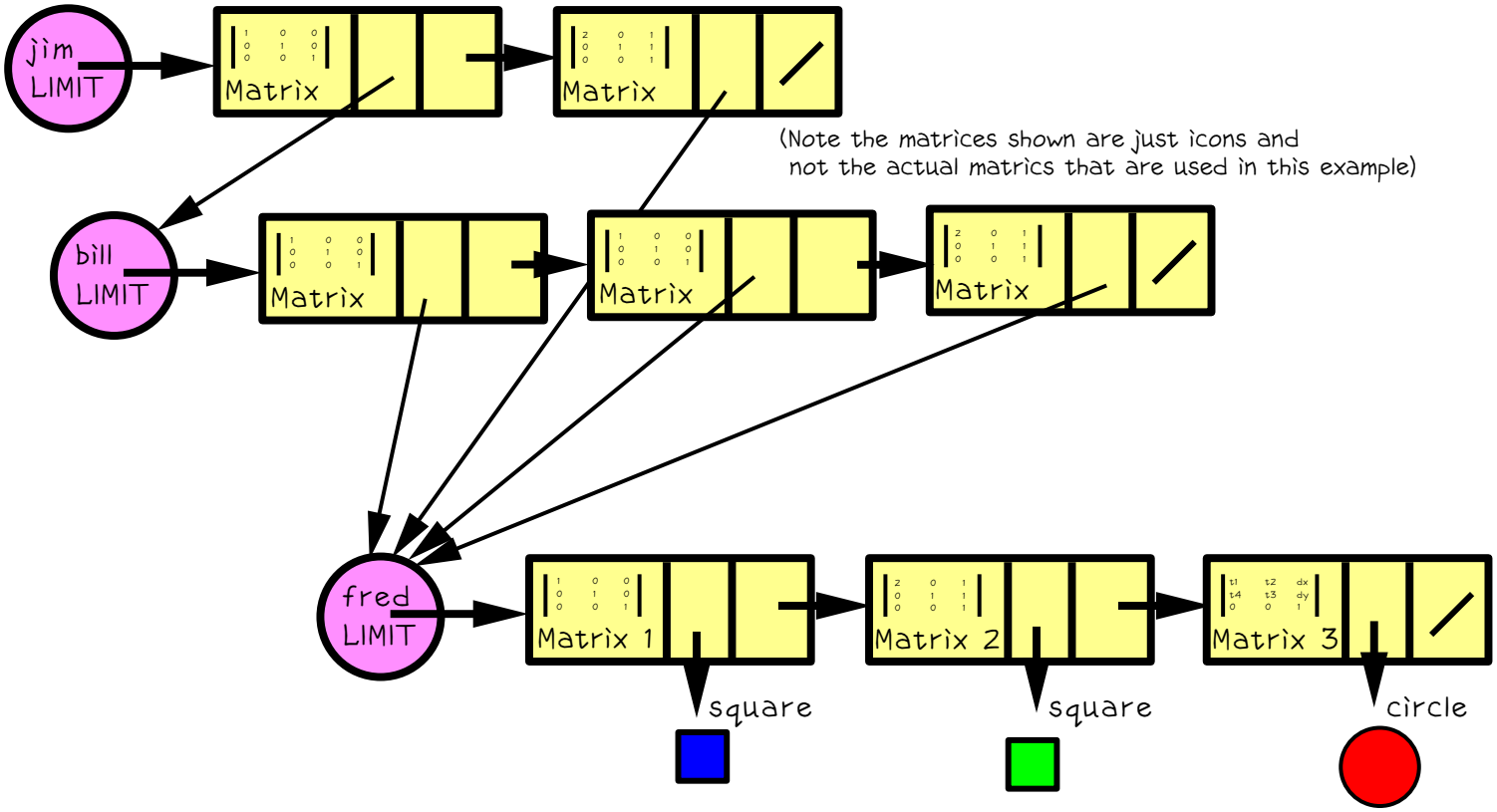
3. NEXT in linked list.

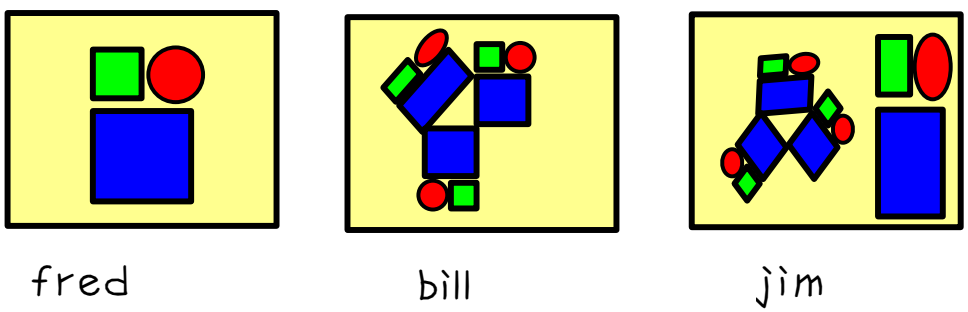
PRIMITIVE

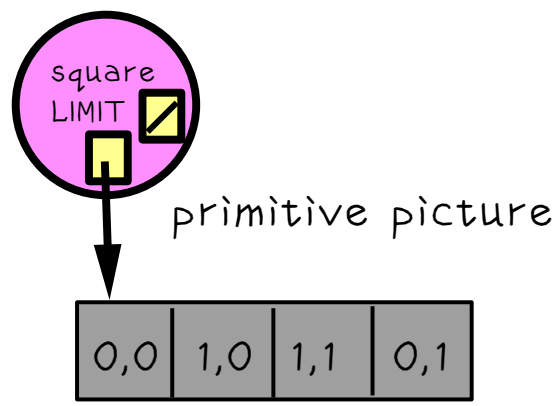
There is a single built in primitive, an array of straight line segments.



Example

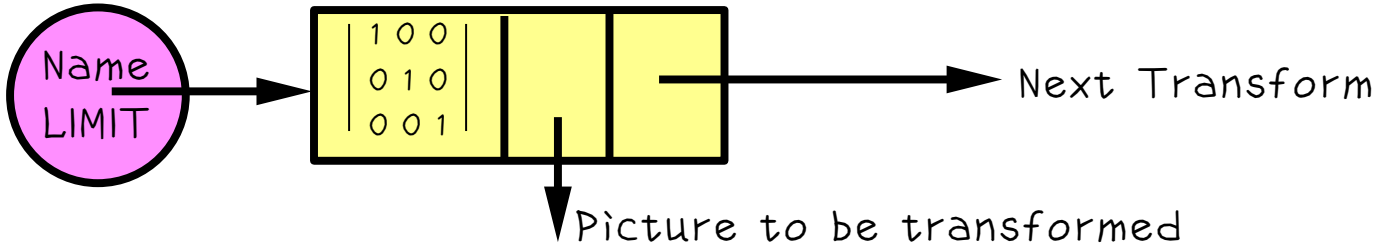

jim
LIMIT
Matrix
Matrix
(Note the matrices shown are just icons and
not the actual matrics that are used in this example)
bill
LIMIT
Matrix
Matrix
Matrix
fred
LIMIT
Matrix 1
Matrix 2
Matrix 3
square
square
circle


fred
bill
jim

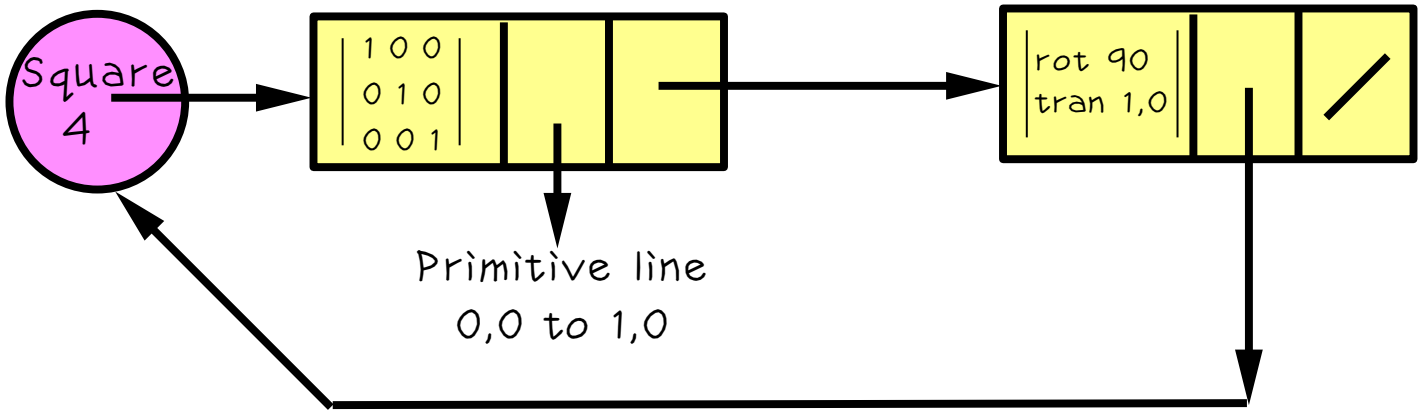

square
LIMIT
primitive picture
0,0
1,0
1,1
0,1

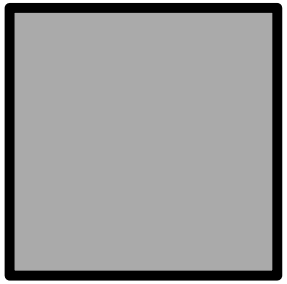
Recursive Hierarchies

Hierarchical Data Structure

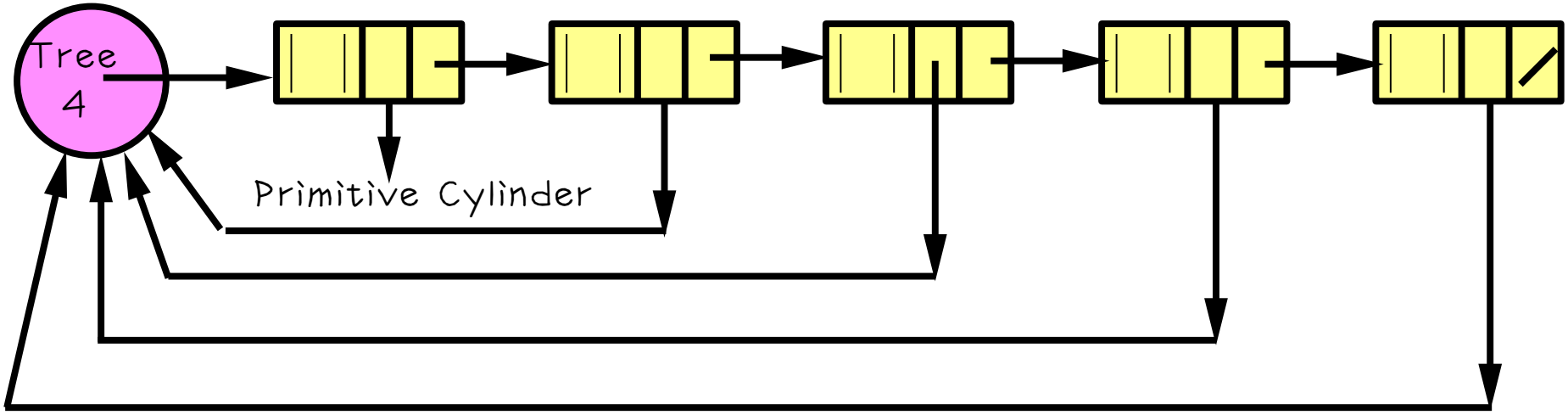


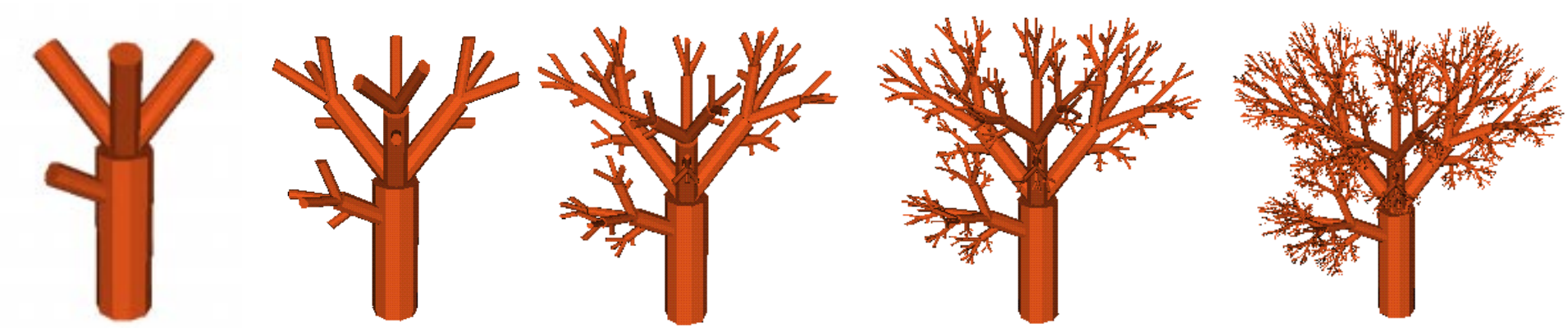
E.g. Square





Multiple Recursion (Tree)



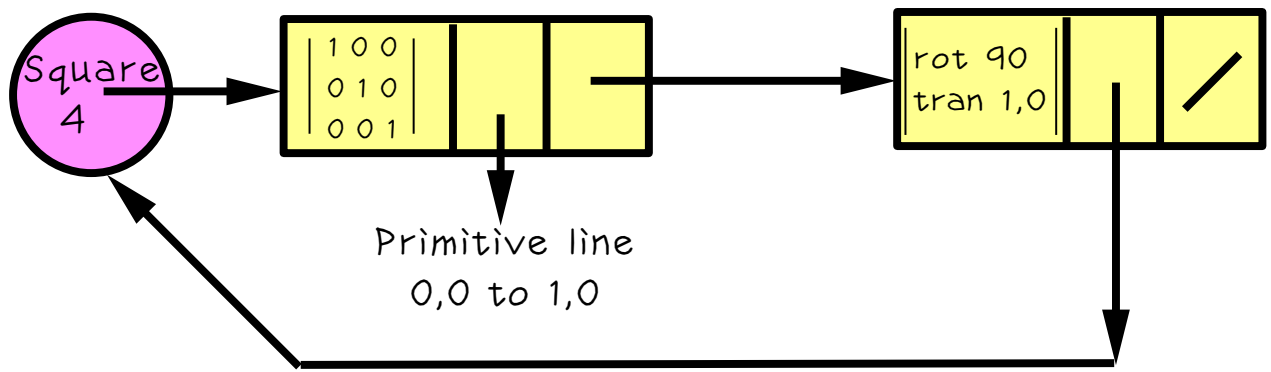


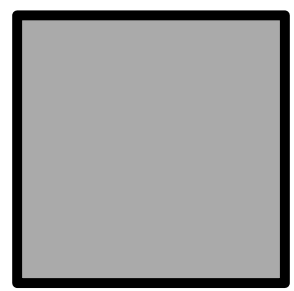
Tree limit 2 limit 3 limit 4 limit 5 limit 7

Traversal algorithm

(pseudo-code)
// matrix m is set to identity for scaling pass and to a scale and
// translate matrix for the window transform

```
plot(PICTURE* p, MATRIX m)
begin
  if (p IS  PRIMITIVE) output(p, m);
  else
   begin
      TRANSFORM t;
     t=p.firstTrans;
     if (p.limit>0)
        begin
          p.limit--;
          while (t != NULL)
             begin
                   plot(t.pic, matmult(m, t.m) );
               t=t.next;
             end  // while
           p.limit++;
         end // if
    end
end
```





Note that Transform objects are initialised with an identity matrix.

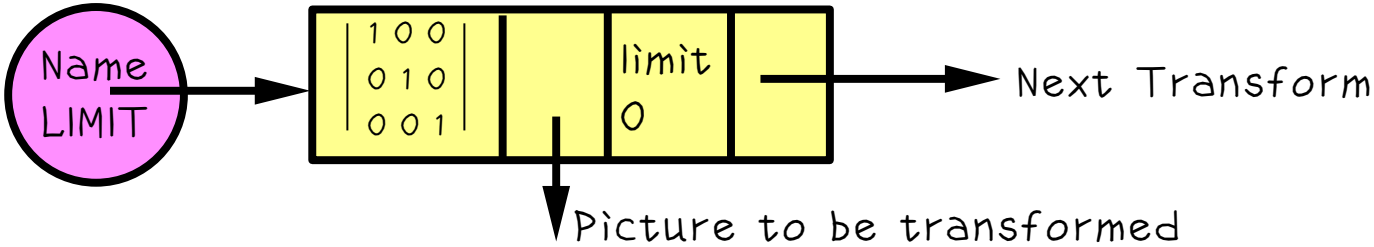
Local Limits

More on the Traversal algorithm

```
plot(PICTURE* p, MATRIX m)
begin
  if (p IS  PRIMITIVE) output(p, m);
  else
   begin
     TRANSFORM *t;
     t=p.firstTrans;
     if (p.limit>0)
        begin
          p.limit--;
          while (t != NULL)
             begin
                if ( p.limit == t.limit) plot(t.pic, postmult(m, t.m) );
               t=t.next;
             end  // while
           p.limit++;
         end // if
    end
end
```

Tree with leaves added but no local limit

Tree with leaves added using a local limit set to zero



Space Filling Curves

The Koch curve:

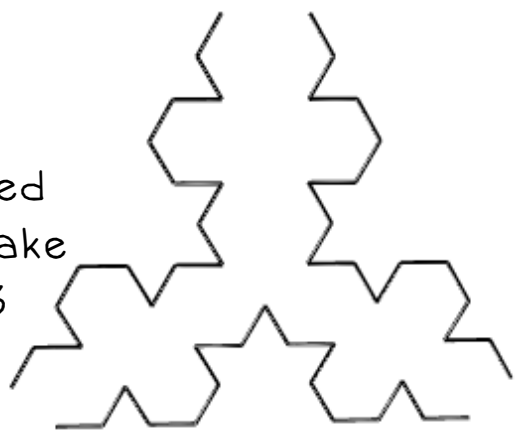
```
o koch
 lim 2
 a line scale 3 if 0
 a koch  scale 0.333333
 a koch  scale 0.333333  rot z 60  ori 1 0 0
 a koch  scale 0.333333  rot z -60 ori 1.5 0.8660254  0
 a koch  scale 0.333333  ori 2 0 0
 col black
c
o snow
 a koch
 a koch rot z 120 ori 3 0 0
 a koch scale 1 -1 1     rot z 60
c
```

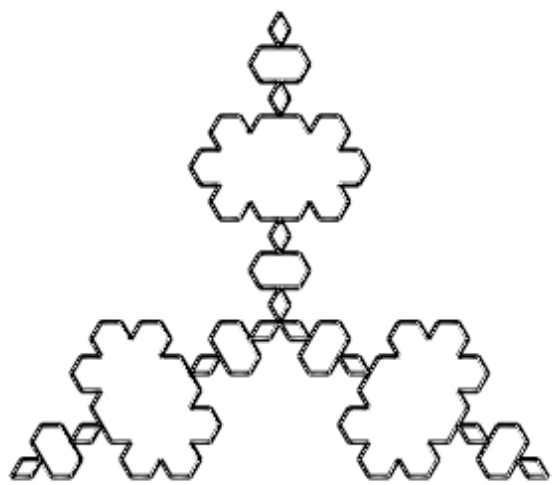
Limit 1

Limit 2

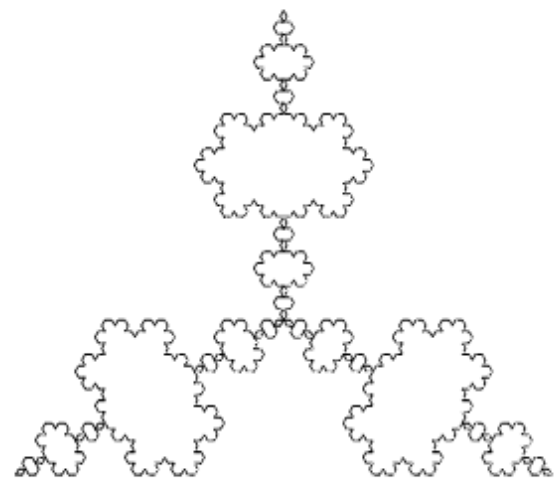
Limit 2

Exploded Snowflake Limit 3





Snowflake Limit 4



Snowflake Limit 5

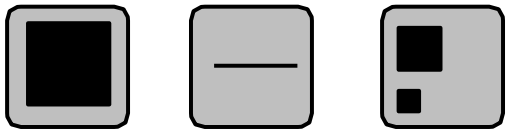
Data Format

commands:

```
# comment for rest of line

 o name          #Open picture definition
 a name          #names can be fixed length e.g. max 6 chars.
# followed by arbitrary number of transformations
   t <x y>       #translate by x,y
   r <theta>     #rotate by theta
   s <x y>       #scale by x,y
   l <number>    # limit <default 1>

   a name
   etc. until ...
 c               #close picture

 d <name>        #define primitive picture
  x y
  x y
  etc.
 c               #close
 w  <xl yb xr yt>  #window limit xleft ybottom xright ytop (for bonus marks)
 p               #plot picture (invoke the traversal algorithm)
```

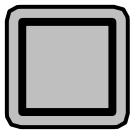
e.g.

```
 d square
   0 0
   1 0
   1 1
   0 1
   0 0
c
o two
   a square
   a square
   s 2 2
   t 0 2
c
d line
   0 0
   1 0
c

p square
p line
p two
```

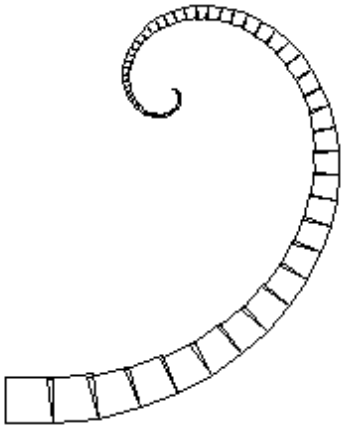


Examples

```
o box
  line
  box
  r 90
  t 1 0
  l 4          # note
that limit (4 in this
case) restricts self
                #
referencing pictures
c
```



```
o spiral
  box
  spiral
  s 0.95
  r 7
  t 1 0
  l 75
c
```



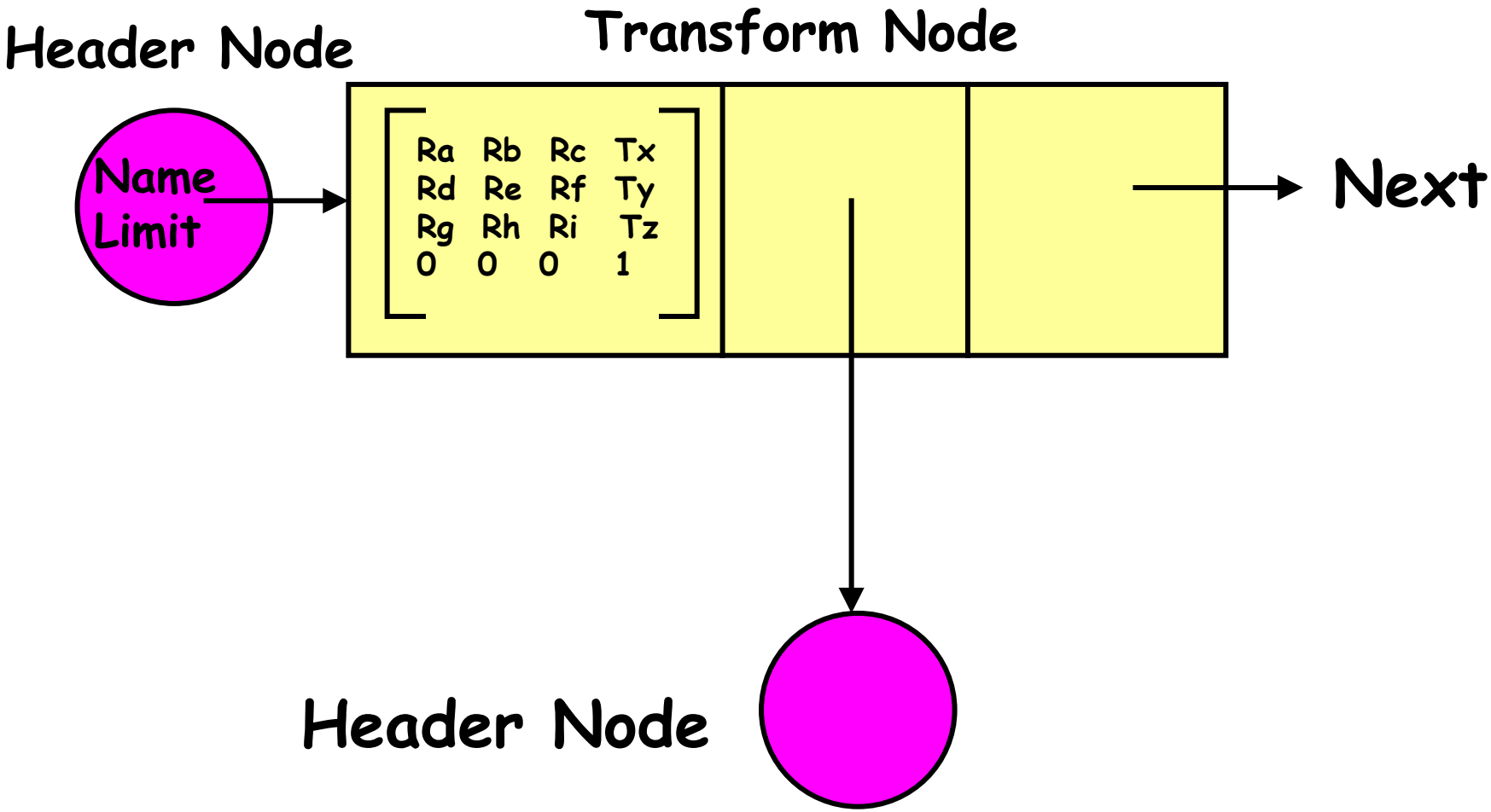
Assignment Two

Read in Model Objects (polygons) in .obj format. (A common format)

Apply user specified transformations to place the objects in a hierarchical data structure.

View with 3D viewing routines (e.g. OpenGL calls)

Bonus: add recursive objects (use recursion limit)


Header Node
Transform Node
Name
Limit
Ra Rb Rc Tx
Rd Re Rf Ty
Rg Rh Ri Tz
0 0 0 1
Next
Header Node

Assignment (continued)

Traversal:

Scaling Pass

1. start with the Identity matrix on the stack.

2. when a TRANSFORM is entered pre-multiply top of stack by M and push result on top of the stack.

3. when a PRIMITIVE is entered apply the top of stack to the line segments. Record max/min limits of the drawing.

4. Set the SCALING matrix according to max/min
The min/max could be set in the input file to overide the automatic scaling pass.

Drawing Pass

as above only start the stack with the SCALING matrix and on step 3 above output the scaled line segments to the output routine.