Proposal for an Animation Language

Seth Steinberg
7/12/76

Data Types

Number = ±6E-67 → ±6E66   ~0 → F else T

Point = X: Number
Y: Number

Color = I: Number   R: Number
H: Number   or   G: Number
S: Number   B: Number

Item = Color: Color
Center: Point
Shape: Point, ooo

Group = nth: Item

Terminal Nodes

Screen ← any number of Items or Groups

any number

T = 1.0   PI = 3.1415926535 ooo

F = 0.0

any "color" from current matrix   or consed up otherwise

Frame → frame number

Time → current time

Functions

F(Number) → Number

sin cos log exp - atan ¬

F(Number, Number) → Number

+ - * / ↑ atan2 & |

before until

F(Number, Number, Number) → Number

between except proportion

F(Number, Anything, Anything) → Anything

select

F(Number, Number) → Point
pointrect pointpolar

F(Point) → Number
xcoor ycoor theta length

F(Point, Point) → Point
midpoint reflect + −

F(Point, Point, Number) → Point
rotate scale

F(Point, ooo) → Item
item

F(Item, Number, Number) → Point
interpolate

F(Item, Number) → Point
point

F(Item) → Point
center

F(Item) → Color
color

F(Item) → Number
npoints lengthit

F(Item, Number) → Item
scaleit1 delayit

F(Item, Point, Number) → Item
scaleit rotateit

F(Item, Color) → Item
colorit

F(Item, Item) → Item
merge reflect* average

F(Item, Point) → Item
positionit

F(Item+, Item+, ooo) → Group
group

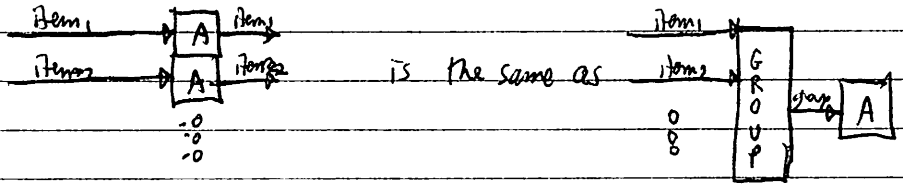
F(Point, Point, ooo) → Group
group

F(Item, Item, Number, Number) → Item
fillin

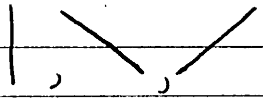
* ~~second~~ npoints (Item$_2$) = 2 only

+ or Group

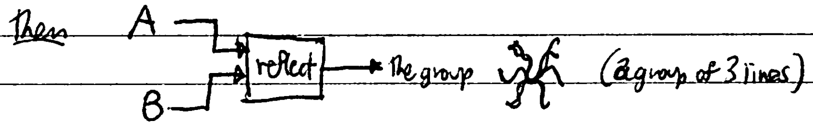
Groups are used to express iteration, but otherwise are treated as Items. So:

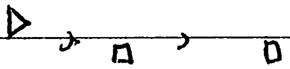


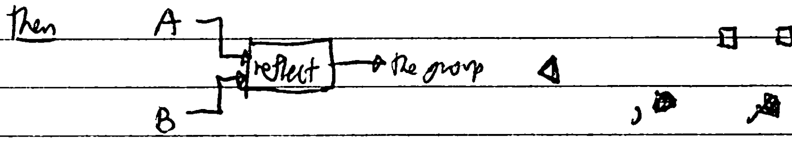
Think of it as ribbon cable.

So if A is the group:  (3 lines or items)

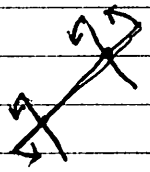
Then if B is:

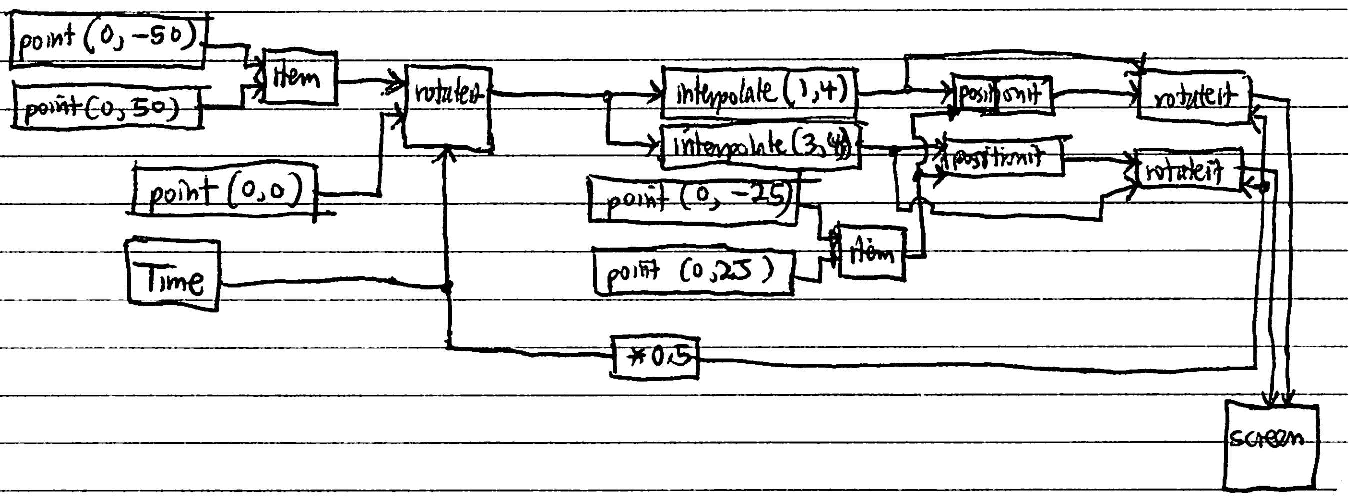


if B is the group: 

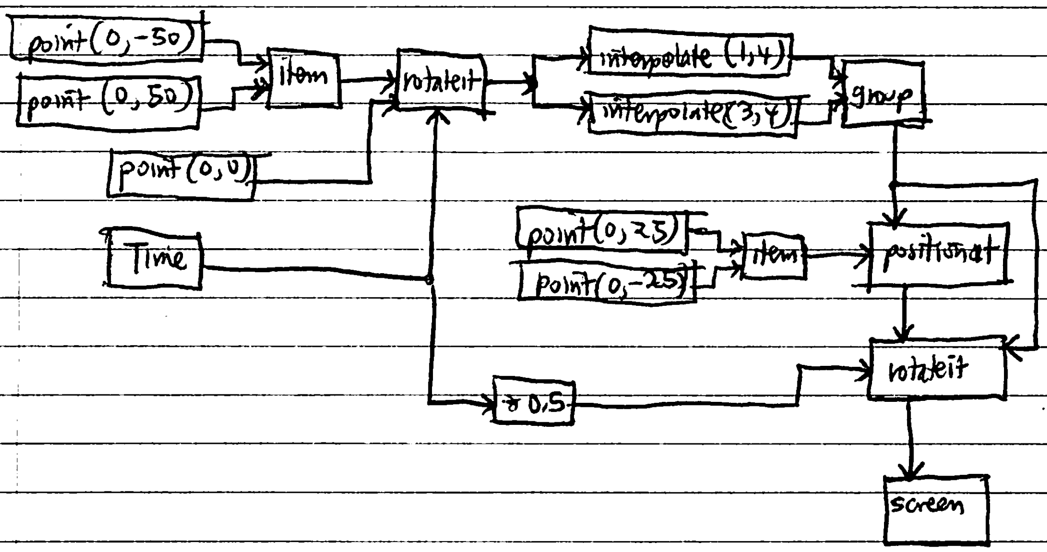


Note that group does not work on groups, only items. If a Group is passed to group, its member Items will be members of the resulting Group.

Example:  2 level nested rotation



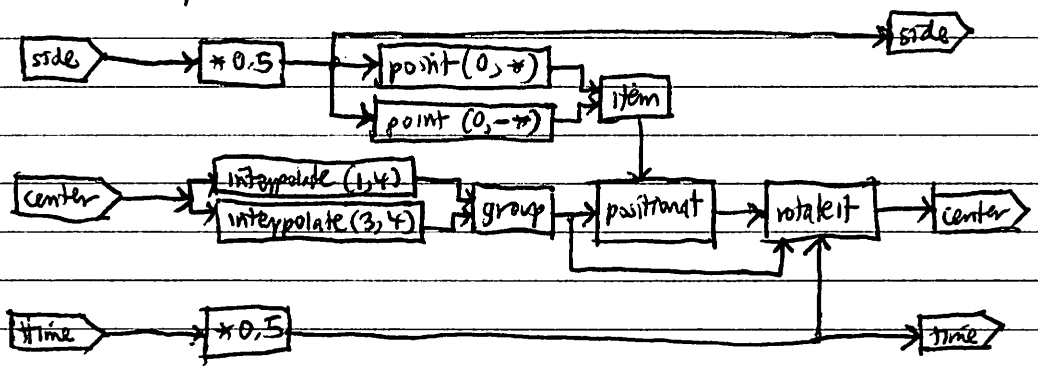
or



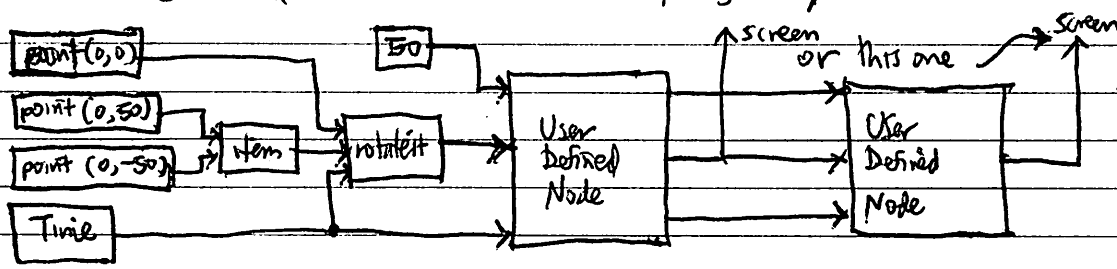
Function Definition or User Defined Nodes

Any network can be saved as a node, presumably in some file. To facilitate the building of useful nodes, a ~~there~~ are a special class of sources and sinks which may have arbitrary labels, which must not conflict with other labels.

For example:



could be used in the above example, thusly:



etc.

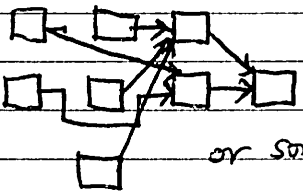
Notice that the center output of the user defined node is a group of twice the size or the center input.

User Interface

1) Node menu : the 20 or so most popular nodes, builtin or user defined.

2) Selector menu : whenever a line is drawn from something, a menu of output selectors is offered; similarly for input selection

3) Node display : probably 2-4 levels deep, in a stylized fashion e.g.



or something

4) Editing functions :

a) change node menu
b) link, unlink
c) create, delete node
d) follow input, follow outputs to move around network

5) Modes :

a) edit
b) run
c) debug
d) draw
e) color editor

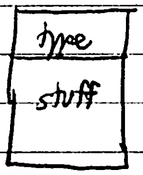
? Xerox mode? A bit map type could be added, perhaps, only with some operations supported?
? How about a color matrix sink??

interpolate	can be used to find mid-points or follow paths
fillin	performs a standard linear key frame interpolation
average	≡ fillin (. , . , 1, 2)

Notes on Implementation

1) It should be obvious that colors are allocated as they are drawn, since they may change freely.

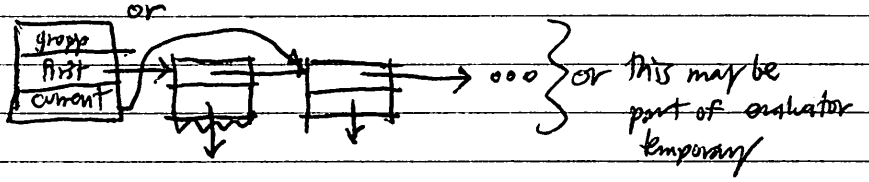
2) value structure



all points in floating point

Value	float
Point	float, float
Color	float, float, float
Item	Point, Color, Point-list
Group	next : ptr, entity : ptr

or



or this may be part of evaluator temporary

Nodes

next in node chain
type, or user defined
ptr to definition
sources
result

3) The null group can be used to indicate no change in drawing or a blank drawing. Perhaps the screen can have a draw enable flag.

Suppose we only want to hack the colors of items. Obviously each item on the screen could be associated with a color slot:
We need: a) only change color mode
b) new slot mode so each item can get a new color

4) Most code should be in PL/I. It would be nice to fit it all in one loadmod, but I doubt it. We shall see.