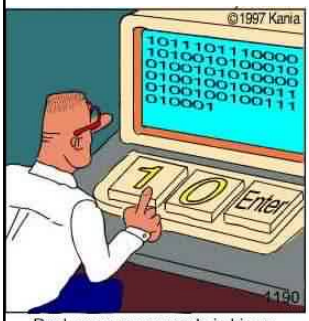
Software Visualization



Real programmers code in binary.

Lecture
WS 02/03

Algorithm Animation Systems

POLKA

- general purpose animation system
- well-suited to building algorithm animations and animations of parallel computations
- easy to use and powerful
- programmers need not to be computer graphics experts to develop own animations
- programmer has to use C++

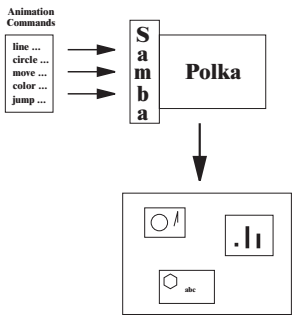
SAMBA

- front-end to POLKA
- interactive animation interpreter
- usable with **any** programming language
- versions exist for Unix, Windows and Java
- batch mode

SAMBA

- designed to be very easy to learn and to use
- reads ASCII commands, one command per line, and performs the corresponding animation action using POLKA

Architecture for SAMBA



Commands

- each command consists of multiple fields
- first field defines the type
- different groups of
  1. create objects
  2. modify objects
  3. alter views

```
line li  0.1  0.1  0.2  0.2  green thin
circle 27  0.8  0.7  0.1  red solid
move li  0.5  0.6
color 27  blue
jump 27  0.3  0.4
viewdef MainView 600 600
view MainView
```

Concurrency of Commands

- explicit concurrency of animation commands with { and }
- commands in braces will be performed concurrently

Developing Algorithm Animations

- Interesting Events
- annotate the implementation with „print“ statements
- program execution builds a trace of Samba commands
- Forward output trace to Samba
  1. Interactive: % algoanim | samba
  2. temporary file: % algoanim > tracer
     % samba tracer


```
int A[10];
double width, stdheight;
sort() {
int i,k,x;
init();
for (k=1;k<10;k++) {
    x=A[k];
    i=k-1;
    while (i>=0 && A[i]>x) {
        A[i+1]=A[i];
        i--;
    }
    A[i+1]=x;
}
```



```
init() {
 A[] = {5,2,7,9,4,8,3,1,10,6};
 double xpos,ypos,xsize,ysize;
 width=1/12; stdheight=1/24; ypos=stdheight; xsize=width-0.01;
 for (int i=0;i<10;i++) {
    xpos=width*(i+1)+0.005;
    ysize=stdheight*A[i];
    printf(„rectangle %d %f %f %f %f blue solid\n“,i,xpos,ypos,xsize,ysize);
 }
 printf(„triangle tri %f %f %f %f %f %f green solid\n“,
      width*1.25,stdheight/3,width*1.75,stdheight/3,width*1.5,stdheight/3*2);
 printf(„pointline line %f %f %f %f red thin\n“,width*2,0,width*2,stdheigh*12);
}
```



```
int A[10];
double width, stdheight;
sort() {
int i,k,x;
init();
for (k=1;k<10;k++) { printf(„moverelative line %f %f \n“,width,0.0);
    x=A[k];          printf(„moverelative %d %f %f\n“,k,0.0,0.5);
    i=k-1;           printf(„move tri %f %f\n“,(i+1.5)*width,stdheigh/2);
    while (i>=0 && A[i]>x) {
        A[i+1]=A[i]; printf(„moverelative %d %f %f \n“,i,width,0.0);
                     printf(„swapid %d %d\n“,i,i+1);
        i--;         printf(„moverelative tri %f %f\n“,-width,0.0);
    }
    A[i+1]=x; printf(„move %d %f %f \n“,i+1,width*(i+2)+0.005,stdheight);
}}
```