ANIMAL [*] User Guide

Dipl.-Inform. Guido Rößling
FB 12, Parallel Systems
University of Siegen
Hölderlinstr. 3
D-57080 Siegen

EMail: `roessling@acm.org`

April 6, 2000

Contents

[*] Covers ANIMAL Release 1.4

1 Introduction

ANIMAL is a compact, efficient and easy to use animation tool developed at the University of Siegen starting in 1998. The motivation for developing ANIMAL came from a lack of satisfying tools for animations that are both easy to generate and edit and helpful in lectures.

ANIMAL is an acronym for A NEW INTERACTIVE MODELLER FOR ANIMATIONS IN LECTURES, which already shows the main area of usage for ANIMAL.

ANIMAL's strengths lie mainly in the following areas:

- very easy to use graphic interface incorporating *drag and drop*,
- animations can be automatically generated using either the scripting language ANIMALSCRIPT or the ANIMALGENERATOR API,
- platform neutral implementation in Java,
- high flexibility.

On first using ANIMAL , the program might appear somewhat limited in the animation effects as well as the primitive graphic types offered. Other programs seem to offer a far more operations at first glance.

However, a closer look will quick reveal that this impression is not necessarily accurate. Some other programs tend to offer the same basic effect not as a configurable effect, but rather as a collection of effects. ANIMAL offers highly flexible, adaptable effects that can usually perform the same other animation programs also offer - although the list of "supported effects" at first seems much shorter. Check out section 4 on page 7 for an in-depth description of the basic effects available and what operations can be done by using them wisely.

Finally, the high flexibility mentioned above comes from the simple fact that practically all operations and objects ANIMAL uses are easy to edit and adapt to specific needs. Furthermore, the available object or animation effect types are not coded directly into the implementation. Thus, developers should find it very easy to add extensions to ANIMAL without having to touch much existing code.

Using the new additions to ANIMAL, it is very easy to generate interesting animations to enhance viewers' motivation and understanding of the topics presented.

ANIMAL is freely available on the *World Wide Web*, complete with a extensive collection of animations including screen shots and description. Check out the official ANIMAL Home page at **Home Page**
`http://www.informatik.uni-siegen.de/~roesslin/Animal/index.html`
for more information.

2 Requirements

Java

ANIMAL is implemented completely in Java. This means several things, one of which being that the performance is - alas - at times somewhat slower than one might wish.

It also means, however, that the *hardware / software* requirements for using ANIMAL are very easy to specify: your computer must be able to run Java. This is something that practically *all* current computer models can do, although with a wide difference in performance. We recommend a moderately modern computer with a "enough" RAM, since *Java* itself is somewhat memory-intensive on most platforms. For example, ANIMAL runs smoothly on my laptop (Pentium II Mobile, 300 MHz, with 64 MB RAM).

JDK: 1.1.5+
Swing 1.1.1

As stated above, ANIMAL is implemented completely in *100% Pure Java*, using SUN's *JDK 1.1.5* or above. The only further requirement is Swing , the graphic package collection that provides a far more attractive alternative to Java's original *Abstract Windowing Toolkit (AWT)*. ANIMAL is currently only available for the current Swing 1.1.1 release also included in JDK 1.2+.

Tested platforms

ANIMAL has been developed both on Digital AlphaStation running Digital OSF v4.0 – also known as Digital Unix – and Intel-based computers running Linux, FreeBSD, Windows 95 / 98 and Windows NT. It has also been tested on an Apple Powerbook and thus underscores Java's claim of "write once, run everywhere".

JDK support

Note that changes in the class libraries, especially the renaming of the Swing packages, prevents us from providing support for both *JDK 1.1.5+* and *JDK 1.2 / 2.0* with the same **source** code. The binary Java code for JDK 1.1.5+ will run under JDK 1.2, so this is the main release we support

3 Starting and Playing Animal Animations

Animal's animation player is included in Animal and is also useful for checking the animation currently under development.

Starting

To start the player, simply type the following command:

```
java animal.main.Animal
```

If this leads to the following message

```
Can't find class animal.main.Animal
```

CLASSPATH

you must update your CLASSPATH environment variable first to include the Animal archive file Animal.jar. This is usually done by locating the place where CLASSPATH is set and appending an entry like

```
C:\Java\Animal.jar
```

for Windows users, or

```
$HOME/Java/Animal.jar
```

for users of Unix-based operating systems. If you are unfamiliar with how to do this, take a look at the Java documentation which should including information about setting the CLASSPATH.

Start Player

To start the Animal animation player, choose the entry Show Animation in the Edit menu of Animal's main window shown in figures 3 on page 8. See figure 4 on page 9 for a screen shot of this operation.

The player itself consists of a single window with video player-like functionality, magnification setting and step choice. Figure 1 on the next page shows an example window.

Most of the player is taken up by the *animation canvas* in which the actual animation is shown.

Components

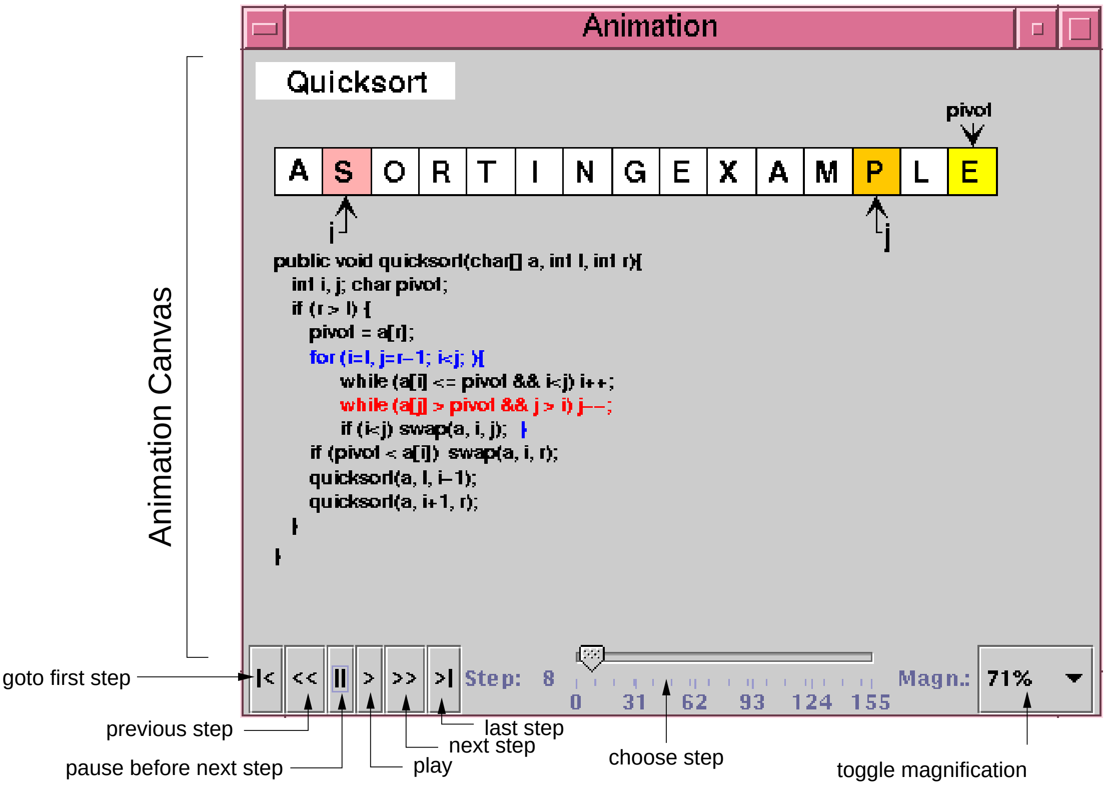
Below this are the controls for the current animation. These consist of a series of buttons which are used to

- jump to the animation's first step $<|$; **Jump to first $<|$**
- go to the previous step *without* playing out the animators contained in the step $<<$; **Previous Step $<<$**
- *pause* before changing the to the next step $||$; **Pause $||$**

  Note that this is only useful if there is a *timed link* between the current step and the next step.
- *play* the current step, ie. activate all animators $>$; **Play step $>$**
- go to the next step *without* playing out the animators contained in the step ensuremath¿¿; **Next Step $>>$**
- jump to the animation's last step $>|$. **Last Step $>|$**

Step Slider

The next component is a *slider* showing how far the animation has progressed. By clicking on the slider icon and dragging the mouse, a "skip forward" effect can be obtained, resulting in a execution of the steps dragged over. The slider can thus also be used to jump to a different step (whether backward or forward) in the animation.



```
public void quicksort(char[] a, int l, int r){
  int i, j; char pivot;
  if (r > l) {
    pivot = a[r];
    for (i=l, j=r–1; i<j; ){
        while (a[i] <= pivot && i<j) i++;
        while (a[j] > pivot && j > i) j––;
        if (i<j) swap(a, i, j);  }
    if (pivot < a[i]) swap(a, i, r);
    quicksort(a, l, i–1);
    quicksort(a, i+1, r);
  }
}
```

Figure 1: ANIMAL Player front end

Note that this may cause problems while generating animations, as the numbers of the animation steps need not be *always* sequential. If you encounter such problems, simply save your animation and reload it, and the problem should be solved.

Magnification

The next component allows the user to select a *magnification* for the display. This is especially helpful for very broad or high animations, grabbing screen shots or scaling the components to allow a switch from *computer presentation* to *beamer presentation* in lectures.

Due to scaling anomalities, only the following “sane” scaling factors are supported:

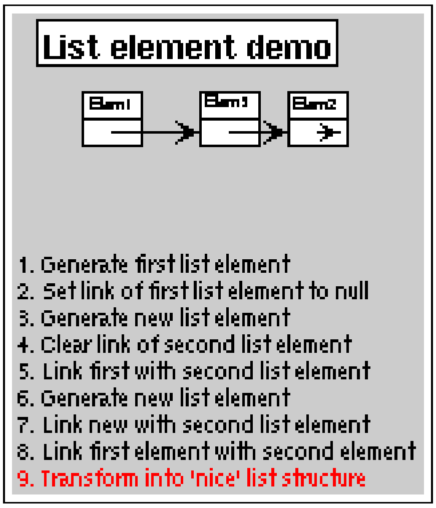
- 50%,
- 71%,
- 100% (default),
- 141%,
- 200%

4 Generating A New Animation

In this example, you will use a few simple steps to generate a short but interesting animation about the behavior of the data structure *singly-linked list*. This animation will illustrate how to use ANIMAL to easily visually build animations.

Result

The final result of this process will look roughly as follows:



List element demo

1. Generate first list element
2. Set link of first list element to null
3. Generate new list element
4. Clear link of second list element
5. Link first with second list element
6. Generate new list element
7. Link new with second list element
8. Link first element with second element
9. Transform into 'nice' list structure

Figure 2: Final result of the tutorial animation

Don't worry, reaching this result is really not difficult. But now, let's get going!

4.1 Preparation for the new Animation

Main Window

First of all, you must start ANIMAL as described on page 5. After a while spent on initialization and loading the initial animation, (at least) ANIMAL's main window is shown. The main window is shown in figure 3 on the next page.

Menus

This window contains menus for *file operations* (`File`), opening and closing the windows (`Edit`) used for editing and viewing the animation, setting the *Options* (menu `Options`), and `Help`. Furthermore, it has a list of buttons which serve as a shortcut for – from left to right – *New Animation*, *Load Animation*, *Input* ANIMALSCRIPT, *Save Animation*, *Save Animation As*. . .

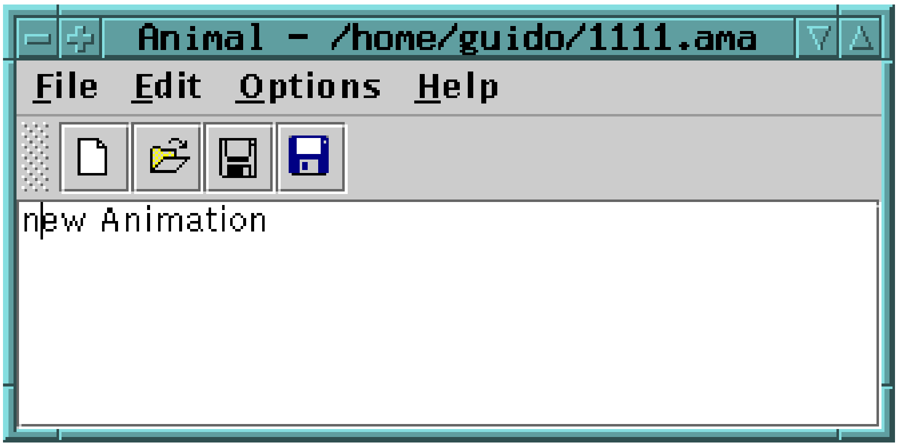
New Animation

For now, you need to create *new animation*, so you should do *either* of the following two operations:

- Click on the first button in ANIMAL's main window showing a *blank sheet*,
- or click on the menu `File` and select its first entry, `New`.

  Shortcuts

  You can also use shortcuts by pressing the shortcut key and the letter highlighted in the menu - in this case, `F`, so press both `ALT` and `F`, and the menu will be



new Animation

Figure 3: ANIMAL's Main Window

displayed. If not, you probably have to replace `ALT` by `CTRL`. If this does not work either, ask your system administrator for the local configuration details.

After the menu is shown, pressing `N` – the letter shown after the entry `New` – is the same as clicking on `New`.

4.2 ANIMAL's Draw Window

Open Draw Window

First of all, you are going to draw a simple object: the *rectangle* underlining the title. To do so, you have to open ANIMAL's *Draw Window*. Go to the *Edit* menu and select the entry *Show Draw Window*, if the window is not already opened. The menu should now have a check mark in from of the entry *Draw Window* as shown in figure 4 on the following page.

DrawWindow

Object Toolbar

ANIMAL's drawing window looks as shown in figure 5 on page 10. At the top of the window, you can see a row of *buttons* for *object generation* - the *Object Toolbar*. Below this row on the window's left are some helpful buttons, the *animation step selection* and a *options* entry. The *status line* at the bottom of the window always displays information about the semantics of the currently selected operation.

Tables 1 on page 11 and 2 on page 12 summarize the buttons shown.

The main part of the window is taken up by the *drawing area* – here showing a snapshot of the *Quicksort* animation. This is the place where all objects are drawn.

4.3 Activating Grid Support

Grid

First, you should activate a *grid* for easier and more precise drawing. Referring to figure 5 on page 10, click on the *pop-down menu* labeled `Grid` and set the value to `20`. Then look for the following button directly below and to the left of the Grid menu:

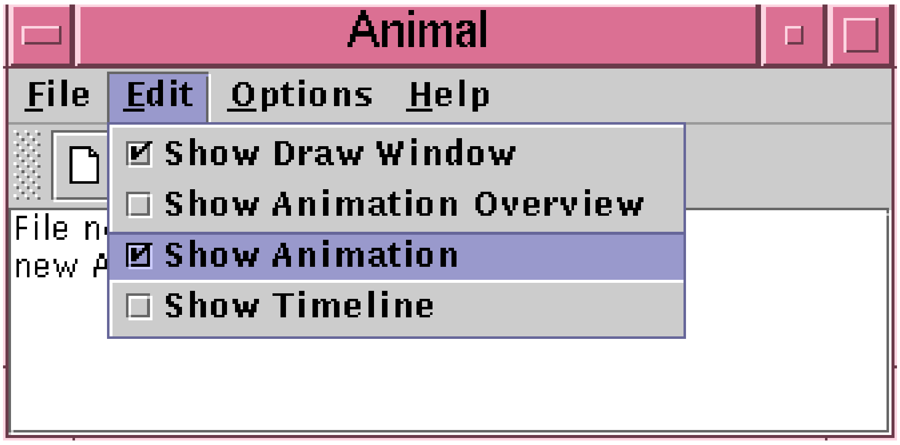


Figure 4: Selecting the displayed windows. Here, both *Animation* and *Draw Window* are opened.

Grid Snap

▦. If the button has a dark grey background, the *Grid Snap* is already turned on; otherwise, click once on the button. This button serves as a *toggle* - each click *inverts* the selection and thus changes from *grid off* to *grid on* and vice versa.

The *grid* is helpful for precise drawing, as it adds a line every *n* pixels in both horizontal and vertical orientation. The exact value of *n* depends on your selection; in this case, the distance between two lines is *n=20 pixels*. By activating *Grid Snap*, you can only draw points falling exactly on those points where two such grid lines meet, and not “in between”.

4.4 Drawing the Header Rectangle

Polyline / Polygon

As the first thing you should draw is the *title highlight rectangle*, click on the symbol for *polyline / polygon* showing a short line: Z.

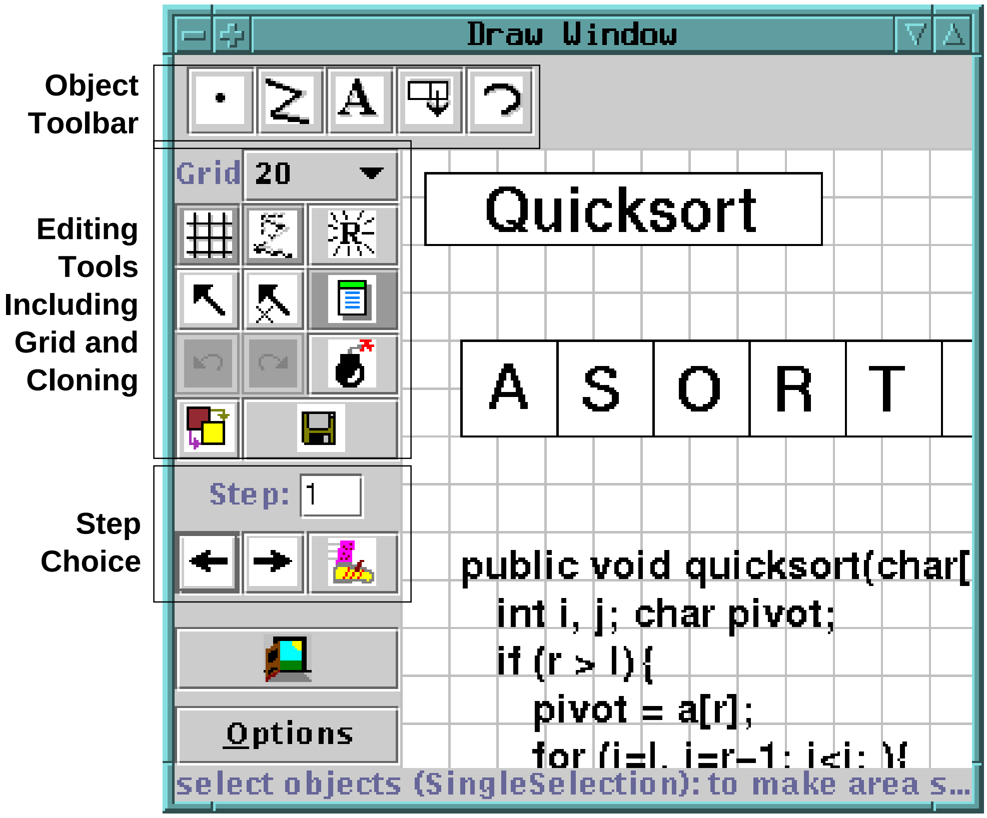
Polyline options

This will cause a window labeled `Polyline Options` to pop up showing one of the displays given in figure 6 on page 13. Move this window out of your way, *but do not close it*.

Polyline drawing

Now, set the *first* rectangle point by clicking on the first point where two of the grid lines meet – the coordinate `(20, 20)`. Now move the mouse to the right over the next **11** vertical lines (to coordinate `(260, 20)`). You should see a line being drawn between the first set point and the current mouse position.

Click the *left* mouse button again to set the second point. Now go down two horizontal lines to coordinate `(260, 60)` and again click the *left* mouse button. Finally, go left until you are at the point directly below the first point and click the *middle* to finish the component. It should now look like a `U` turned by 90 degrees, open to the



Grid 20

Step: 1

Quicksort

A S O R T

```
public void quicksort(char[
  int i, j; char pivot;
  if (r > l){
    pivot = a[r];
    for (i=l, j=r-1; i<j; ){
```

select objects (SingleSelection): to make area s...

Figure 5: ANIMAL's Drawing Window

Button	Function
	Object Toolbar
•	Button for generating a new *Point* object
Z	Button for generating a new *Polyline* or *Polygon* object
A	Button for generating a new *Text* object
⊡	Button for generating a new *list element* object
ↄ	Button for generating a new *Arc, Ellipse, Circle* or *Ellipse / Circle segment* object

Table 1: Buttons in ANIMAL's Draw Window

left. The component is now finished. . . but it is not really a rectangle, as it is still open!

4.5 Setting Object Attributes and Color

Go to the *Polyline Options* window shown in figure 6 on page 13 and click on the entry *Attributes* to bring up *Object Attribute Selection Pane*.

Polyline Options

Closing Objects

Filling Objects

Here, you can set some options for the component. As you need a *filled rectangle,* click once box before the entry `closed` to add a line connecting the first and last node. After clicking on the box, a check mark appears before the entry. Now you have a closed rectangle, but still not a filled one. So, simply click on the entry `filled` which is only active if `closed` is also selected. Now the rectangle is filled.

Color Choice

If the colors are not to your liking, click on the *Color* label in the *Polyline Options* window and select a new color for the *rectangle outline* with the *Polyline:* menu, or a new fill color using the *Fillcolor:* menu. The menu is used just as the *Grid* menu - just click on it to open the menu and select an entry by clicking on it. If the entry you look for is not visible, use the *scrollbars* on the right as shown in figure 7 on page 14.

depth

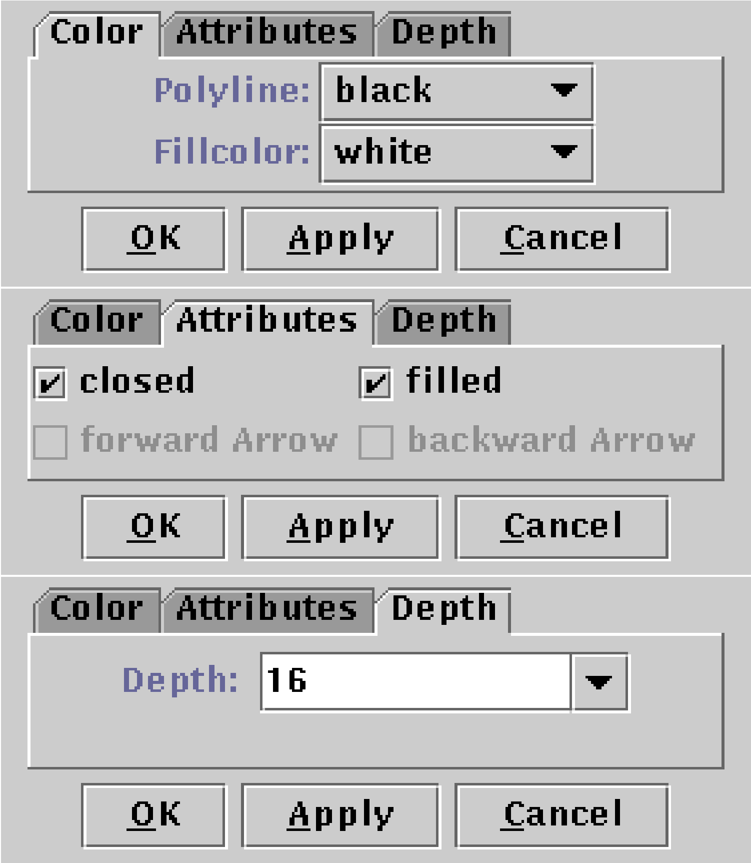
To make sure that the header is placed on the rectangle, and not the other way round, you can set the *depth* of the polygon to a value larger than the one for the text. For now, set the depth to 16 , as shown in the screen shot. The higher this value is, the further to the background ("deeper") the object will be, and will thus be more like to be partially hidden by other objects.

Write Back

When you're done, press the *OK* button in the *Polyline Editor* to close the window. Next, press the *Write Back* button to store the current state of the animation. The button looks like this: 💾

Editing Tools	
Grid 10 ▼	Menu for setting the *Grid* size
⊞	Toggles *Snap* mode on / off: points can only selected at the meeting of grid lines when *snap* is on.
	Toggles the display of temporary objects used for moving other object etc.
R	Repaint the display
↖	Switch to object selection mode
↖	Toggle selection of multiple objects on / off
	Toggle usage of editors on / off
↶	Undo last operation
↷	Redo last undone operation
💣	Delete selected object(s)
	Clone selected object(s)
💾	write back changes to the animation and update windows
Step: 1	Choose step
←	Previous step
→	Next step
	Run step in animation window

Table 2: Buttons in ANIMAL's Draw Window



Polyline: black
Fillcolor: white

☑ closed ☑ filled
☐ forward Arrow ☐ backward Arrow

Depth: 16

Figure 6: Polyline Options Editor for setting *color*, *attributes* and *depth*

4.6 Drawing A Text Component

Text Objects

Now you can add the header text “List element demo” to your current animation. To do so, first click on the symbol for *text* showing the capital letter A:

A

Text Options

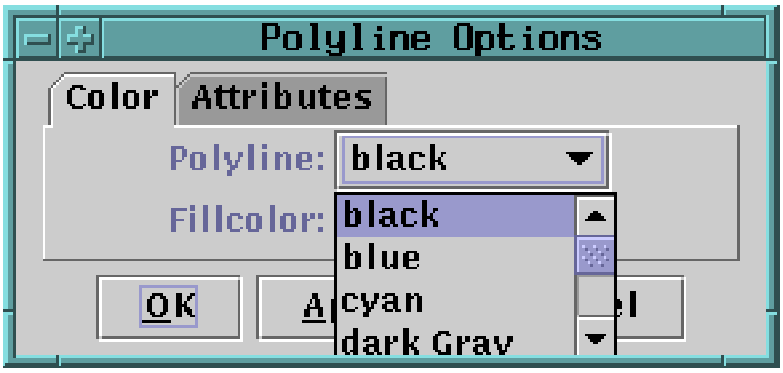
This will open the *editor window* for text components with title `Text Options`, similar to what happened when you clicked on the *polyline / polygon* symbol. This editor window is shown in figure 8 on page 15. The *depth* part of the window is not shown, as this is identical for all objects.

Text Font

First, we are going to set the *text font*. Therefore, click on the *Font* tab, and set the values as shown in figure 8 on page 15 to *SansSerif* font, size *24*, neither *italics* nor *bold*.

Entering Text

For entering the text, click once on the *Text* tab for entering the text itself. Now

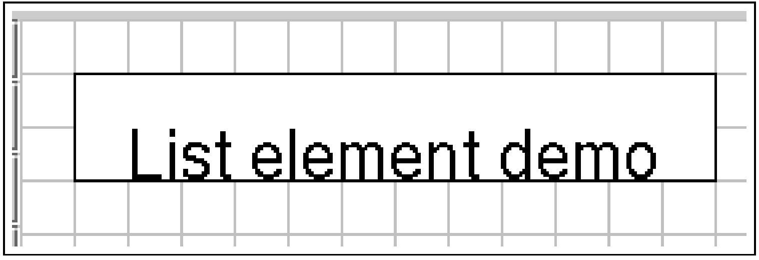


Polyline: black

Figure 7: Color Selection Menu

Placing Text

simply type in the text “List element demo” into the *text field* as shown in figure 8 on the following page. You can also adjust the *text* color as described in section 4.5 on page 11. Place the text inside the *header rectangle* by clicking on the first point at the bottom *inside* the rectangle. Your text should now have 20 pixels space to both the left and right side, and touch the bottom line of the header rectangle as follows:



You can also experiment with the *Font* settings after the text has been placed - just change back to the *Font* tab and see what happens when you click on the *italics* or *bold* check boxes.[1] Before continuing, make sure you have set the entries back to *SansSerif* size *24* with neither *italics* nor *bold*.

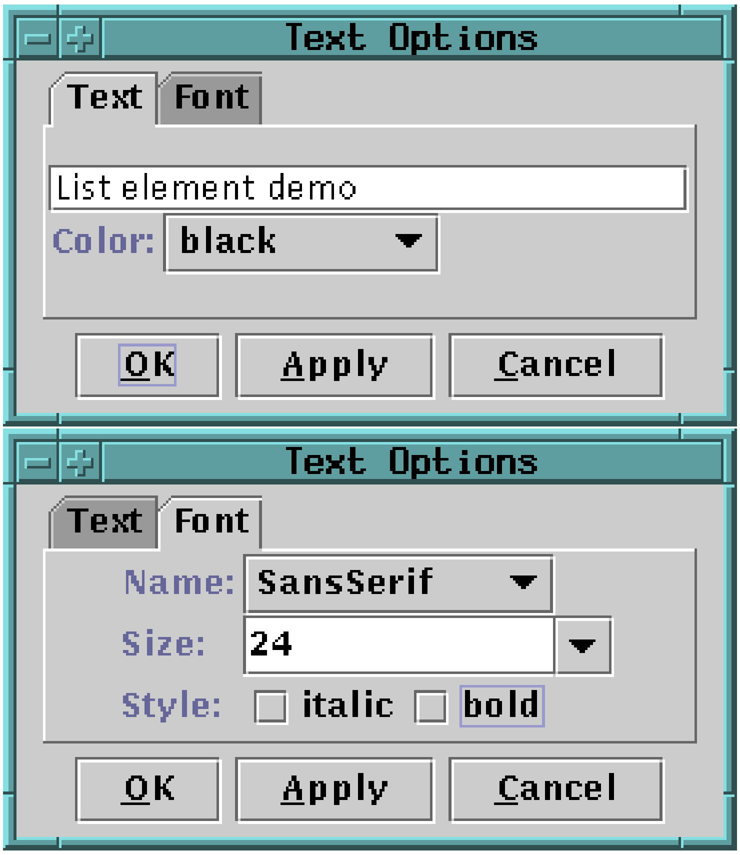
When you’re done, press the *OK* button in the *Text Editor* to close the window.

4.7 Adjusting Object Placement

Grid / Snap

The current display is not very attractive, as the header text has some free space to the left, but none to the right. To change this, you have to *turn off Grid Snap*, since moving the text to the left would only invert the situation: no space to the *left*, but free space to the *right*. Therefore, click once on the *Grid Snap* icon ▦ as described in section 4.3 on page 8 to turn it *off* for now.

[1] Note that some systems may not support SansSerif fonts which are *italics*, **bold** or ***bold italics***. This is *not* a problem caused within ANIMAL’s ability to handle, but reflects the Java installation settings.



List element demo

Color: black

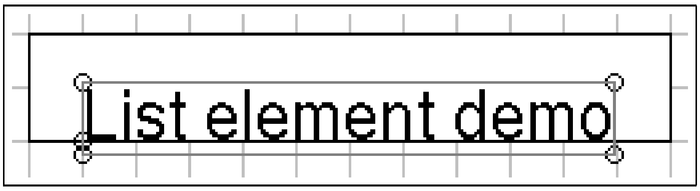
Name: SansSerif

Size: 24

Style: □ italic □ bold

Figure 8: *Text Editor* Window for setting text options

Now, you can click on the text – *anywhere* inside the text. An outline around the text with circles at all edges (two circles at the bottom left) should appear, looking like this:



Dragging Objects

Now, click on one of the circles and *keep the left mouse button pressed*. These circles are called **drag points** and are used for dragging the object along with any mouse movements. So, move your mouse around a bit and see how the text follows the movement.

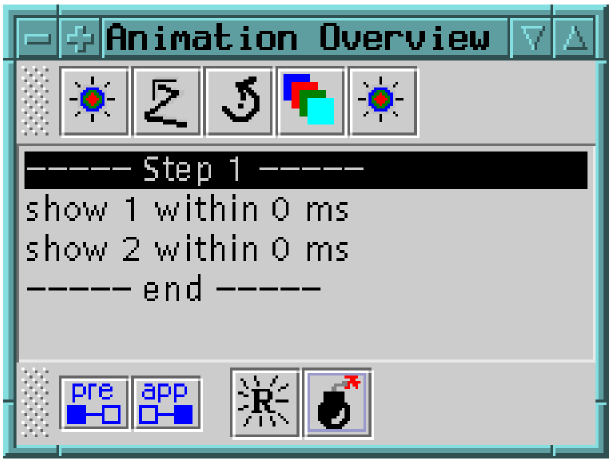
When you try to center the text in this *freehand style* inside the header rectangle, you may find it difficult to place it precisely in the middle. To make this somewhat easier, drop the text somewhere by releasing the left mouse button. Now, turn *Grid Snap* back on as described above. Then, set the *Grid width* to **5** as described in section 4.3 on page 8 and repeat the moving process by clicking on the text and dragging it using one of the *drag points*. You should find it easy to (roughly) center the text now.

4.8 Generating A New Animation Step

The current display containing the centered heading shall be enough for the animation start. Therefore, we need to add a *new animation step* for the next display.

Add New Step

To do so, open the *Animation Overview Window* by activating the entry `Show Animation Overview` in the `Edit` menu of ANIMAL's main window as shown in figure 3 on page 8. The window which opens should look as shown in figure 4.8.



```
------ Step 1 ------
show 1 within 0 ms
show 2 within 0 ms
------ end ------
```

Button Descriptions

At the top and bottom of this window, there is set of buttons. The top button row is used for *adding animation effects*, while the bottom button row offers operations for *animation maintenance*.

The top button row from left to right contains buttons for the following animation effects:

- *show / hide* **without** timing – deprecated, only available for backwards compatibility;

Show/Hide

- *moving* selected objects,

Move

- *rotating* selected objects,

Rotate

- *changing the color* of selected objects, **Color Change**
- and *showing / hiding* selected objects with adjustable timing. **Timed Show/Hide**

The bottom button row contains the following buttons from left to right:

- *Prepend new step* pre **Prepend Step**

  This is used to insert a new animation step *before* the current animation step. Especially useful when you find you need a new step inserted *before* the current first animation step.

- *Append new step* app **Append Step**

  This adds a new animation step directly after the *current* animation step.

- *Redraw* R **Redraw Display**

  This button causes a redraw of the window and is useful when the display becomes muddied.

- *Delete* 💣 **Delete Step/Object**

  This button is used to delete the currently selected entry - either an *animation effect* or an *animation step*.

  In the situation shown in 4.8 on the preceding page, selecting the button – **don't do this now** – would delete the current animation *step*. Of course, a dialog will ask for confirmation before such an operation is actually carried out.

As we want to add a new animation step *after* the current first animation step, click once on the *Append Step* button app. This will lead to the addition of the new animation step **2** and will also directly set this as the current animation step.

4.9 Entering Several Lines Of Text

Now, we are going to enter the documentation for this animation. This consists of the following text entries:

- `1. Generate first list element`
- `2. Set link of first list element to null`
- `3. Generate new list element`
- `4. Clear link of second list element`

- `5. Link first with second list element`
- `6. Generate new list element`
- `7. Link new with second list element`
- `8. Link first with new element`
- `9. Transform into 'nice' list structure`

First, set the *grid size* back to *20* and turn on *Grid Snap* if it is not already turned on.

Entering Text

Now open the *Text Editor* again as described in section 4.6 on page 13. Select either SansSerif or `Monospaced` as the font, size *16*, neither *italics* nor *bold*.

Enter the first text – `1. Generate first list element` – as before by typing it into the text field under the *Text* tab. If you do not recall how this is done, look it up in section 4.6 on page 13.

Entering multiple

Place the text at at the same horizontal position as the header rectangle, but **8** lines below it. *Do not close the Text Editor* window!

Now, enter the second text, replacing the first text in the text field. Do not press **OK** or **Apply**! Position the *new* text one line below the first text. Proceed in the same way with the other lines of text until you reach the state shown in figure 9 on the next page.

Correct Typos

If you have made some typing mistake, you can fix it either

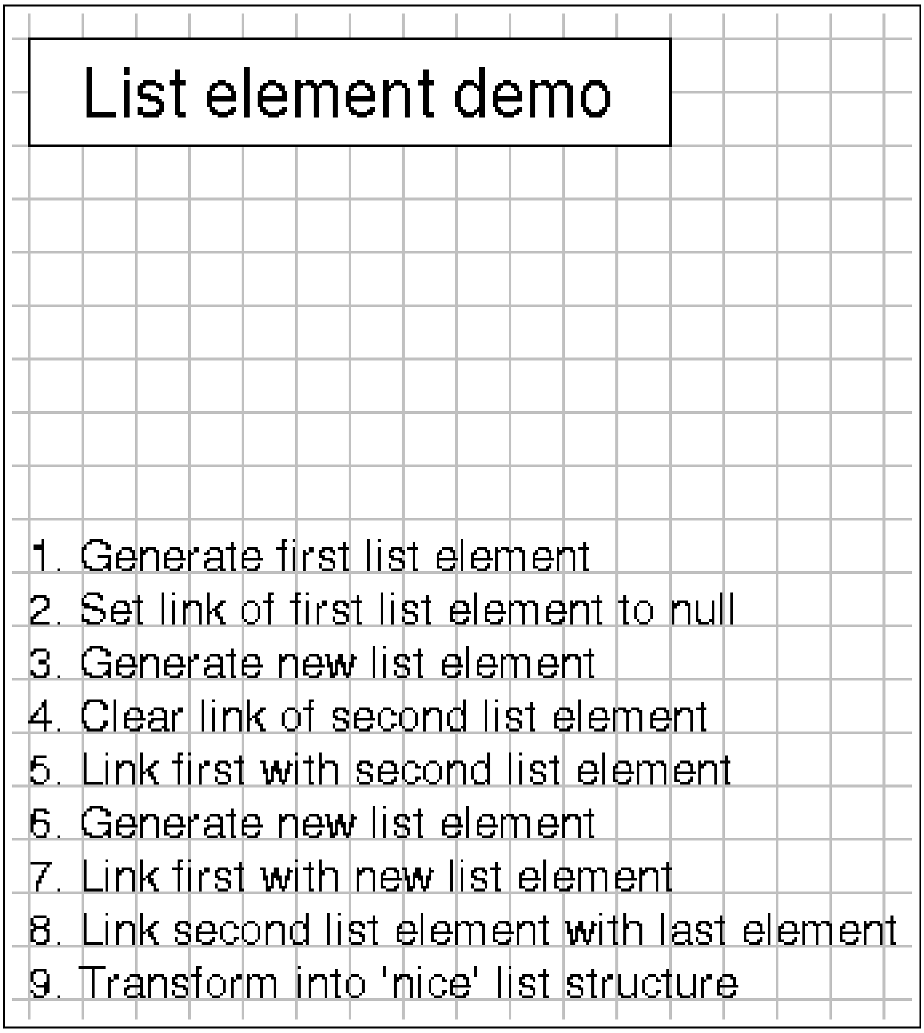
- *before* you have placed another object: simply adapt the text in the text field and press the **Apply** button,
- *after* you placed another object: *close* the *Editor* window after placing the current object, then click on the object in question. If the *Text Editor* does not open, you have to click on the *Editor button* on the left border of the *Draw Window*:

4.10 Storing An Animation

Storing

This is a good time for storing the animation! Animations are stored in one of the following ways:

- Clicking on the *Save* button in ANIMAL's main window ,
- Clicking on the *SaveAs* button in ANIMAL's main window ,
- Selecting `Save` from the `File` menu in ANIMAL's main window,
- or selecting `Save As` from the `File` menu in ANIMAL's main window.



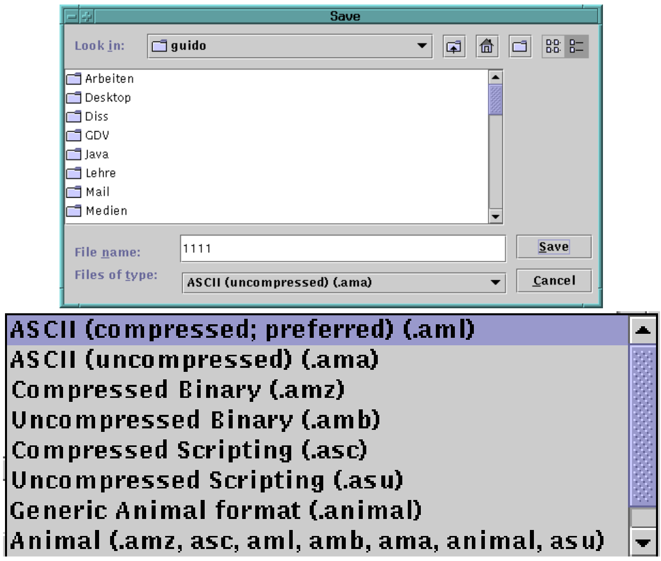
List element demo

1. Generate first list element
2. Set link of first list element to null
3. Generate new list element
4. Clear link of second list element
5. Link first with second list element
6. Generate new list element
7. Link first with new list element
8. Link second list element with last element
9. Transform into 'nice' list structure

Figure 9: State after entering the animation documentation

These components are shown in figure 3 on page 8.

As you have not yet selected a filename for this animation, you will be prompted for a filename *regardless* of whether you chose `Save` or `SaveAs`. The dialog for filename selection looks as shown in figure 10.



Save

Look in: guido

Arbeiten
Desktop
Diss
GDV
Java
Lehre
Mail
Medien

File name: 1111

Files of type: ASCII (uncompressed) (.ama)

Save

Cancel

ASCII (compressed; preferred) (.aml)
ASCII (uncompressed) (.ama)
Compressed Binary (.amz)
Uncompressed Binary (.amb)
Compressed Scripting (.asc)
Uncompressed Scripting (.asu)
Generic Animal format (.animal)
Animal (.amz, asc, aml, amb, ama, animal, asu)

Figure 10: ANIMAL's File Selector. The possible file format selections are shown at the bottom.

Preferred Format

Normally, you will want to store the file in *compressed ASCII* format. Note that this is the first entry in the list and is clearly marked as *preferred*.

4.11 Generating A List Element

Before you generate the first list element, please insert a new step by pressing the *append* button in ANIMAL's *Animation Overview* window as described in section 4.8 on page 16. This should be *step 3*. Note how this addition of a new step also automatically causes ANIMAL to update its *AnimationOverview* window by adding an entry for displaying the text components entered so far.

List Elements

For generating a new list element, select the *List Element* button - also called a *BoxPointer* due to its look. The button looks as follows:

Set Text, Pointer

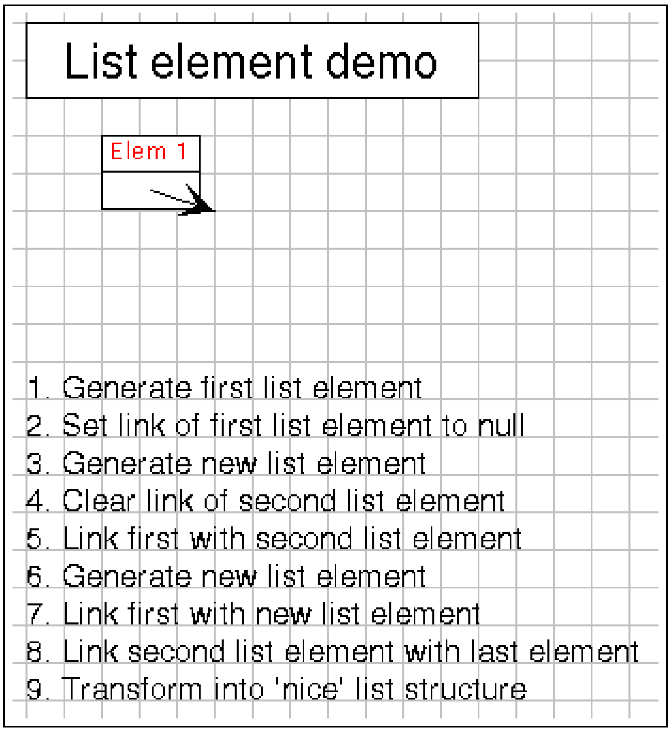
The list element needed has the *text entry* `Elem 1` and *one* pointer. Therefore, select the *Text* tab to enter the text `Elem 1` **without pressing OK or Apply**, then change to the *Pointer* tab to choose the following settings: position *bottom*, *1* pointer.

Placing List Elements

After you have done so, place the list element. The *first* click places the basic object and should place it two 20-pixel squares to the left of the header rectangle box, with 20 pixels space between the element and the rectangle.

Placing List Pointers

The *second* click places the object's pointer, which should point to the next possible point to the lower right of the object. Note how the *status line* at the bottom of the *DrawWindow* tells you exactly what each mouse click means. Your display should now resemble figure 11.



1. Generate first list element
2. Set link of first list element to null
3. Generate new list element
4. Clear link of second list element
5. Link first with second list element
6. Generate new list element
7. Link first with new list element
8. Link second list element with last element
9. Transform into 'nice' list structure

Figure 11: State of the animation after adding the first element

4.12 Highlighting Elements By Color Changing

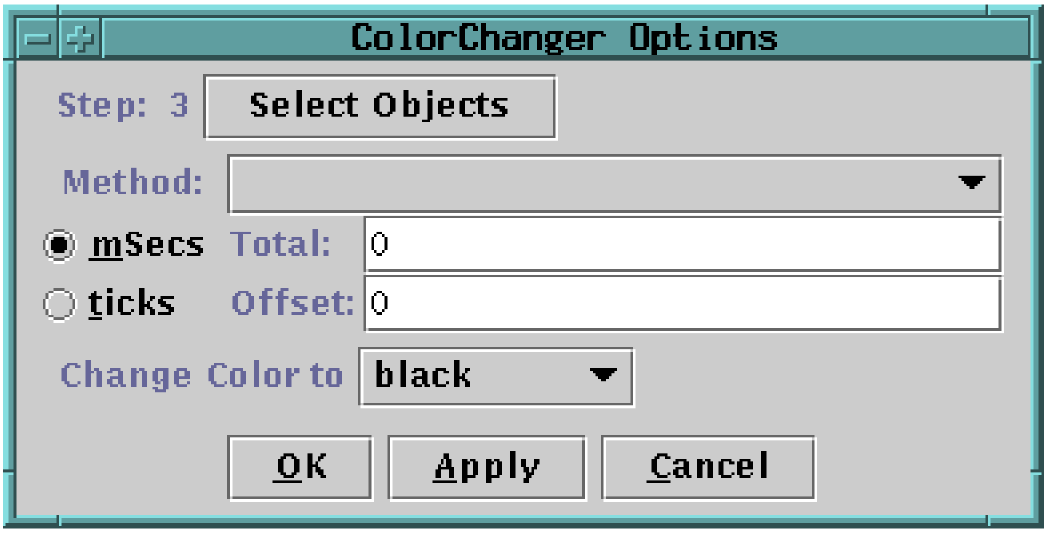
In order to make sure users understand the connection between the new list element and the first instruction, you can change this line to *red*. However, if you do so using the *Text Editor*, you actually change the color of the object *for the whole animation*, which is unwanted in this case.

Color Changer

Therefore, choose the *ColorChanger* button in ANIMAL's *AnimationOverview* window instead. This brings up the *ColorChanger Editor* title `ColorChanger Options`, shown in figure 12 on the following page.

First, click on the *Select Objects* button at the **top** of the Color Changer Editor. The button will now turn dark to show it is active.

Go to the *Draw Window* and click *once* on the first text line. Notice how the entry in the Color Changer Editor changes. The editor allows you to select as many objects for


ColorChanger Options

Step: 3

Method:

● mSecs Total: 0

○ ticks Offset: 0

Change Color to black

Figure 12: *ColorChanger* Editor window

simultaneous animation as you want; for now, the text line is sufficient, though. Click again on the *Select Objects* button, which should now no longer be dark, but display a message like `Selected Objects: 3`.

Method Selection

Next, choose the *animation method* from the list. For a *text* component, this is limited to the default entry *color*, so you do not really have to do anything here.

Delay

Finally, you can decide whether you want to use a *delay* before the object changes color. The *duration* is ineffectual for color changes. Note that you can decide between *ticks* or *ms (milliseconds)*. For *ms*, multiples of `100` make sense. For *ticks*, you can use smaller units, such as `5` or ten.

Set the delay to a short interval, for example *10 ticks*. Finally, choose the *target color*, for example *red*.

Testing

Now open the *Animation Window* by selecting the *Show Animation* entry in the `Edit` menu of Animal's main window (see section 4.2 on page 8 if you are unsure of how to do this). Watch your animation and see what happens in step 3.

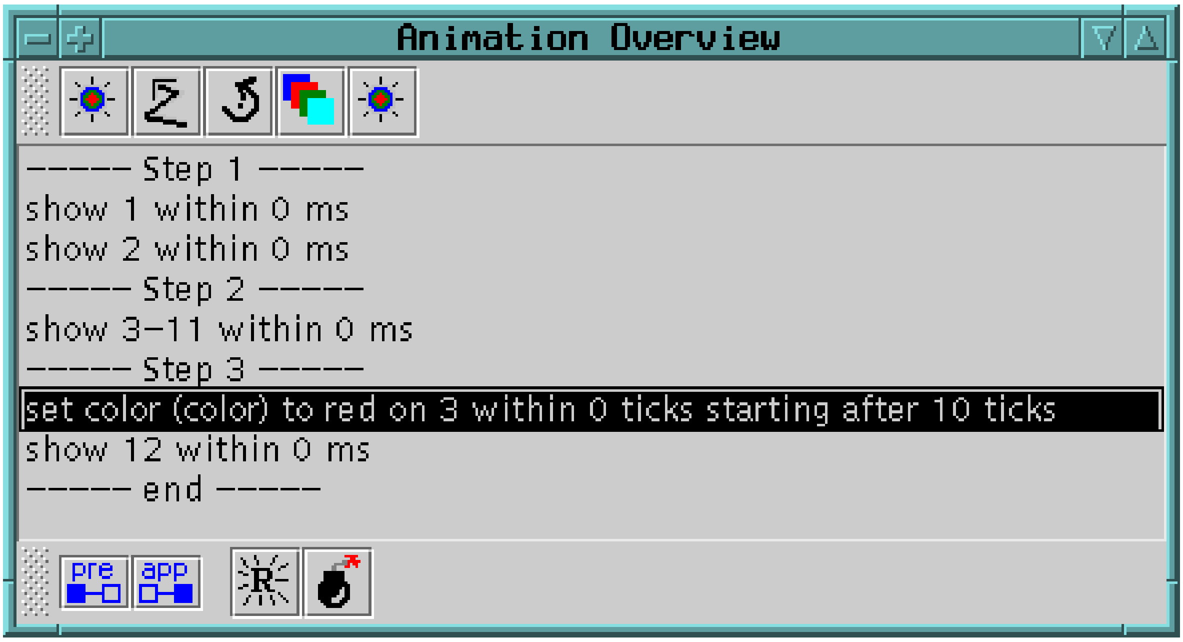
Strange... *first* the element is shown, *then* the line is highlighted! Change this by *double clicking* on the line containing the *ColorChanger* in the *Animation Overview* window as shown in figure 13 on the following page.

The *Color Changer Editor* window should now be open again and allow you to set the *delay* back to 0 and closing the window by pressing **OK**. Next, double-click on the *Show* animator in the same step[2]. You can now assign a delay time to the display of the list element, for example `10 ticks`. Also close this window using **OK**.

Running

Next, press the *Run* icon in the *Draw Window* to re-display this animation

[2] Shown below the selected `ColorChanger` in figure 13 on the next page



```
------ Step 1 ------
show 1 within 0 ms
show 2 within 0 ms
------ Step 2 ------
show 3-11 within 0 ms
------ Step 3 ------
set color (color) to red on 3 within 0 ticks starting after 10 ticks
show 12 within 0 ms
------ end ------
```

Figure 13: Selecting an animator

.

step. Experiment with the delay settings until you are satisfied.

Finally, insert a *new step* for the next effects, containing two color changes and one *move* effect. These operations **cannot** be performed in the same step as the object generation, as ANIMAL only allows you to use **one** animation effect on each object per step. *Displaying* a new element causes the insertion of a *show* animation effect, therefore trying to add a new effect on this element would mean having *two* effects for this element.

In the new step, highlight the second command in *red* using the same steps as described in the last section for generating a *Color Change* effect. You should now have two red texts, which is somewhat unfortunate.

Therefore, you might want to mark the first line of text as "done". To do so, repeat the steps of the last section to enter a new *Color Change* animation effect that sets the color of the *first* line to *blue*.

4.13 Moving Elements

Move Editor

The next operation calls for changing the *tip* (arrow) of the current list element to be set to *null*. *Without* changing the step, click on the *Move Editor* button in ANIMAL's *Animation Overview*. The window that pops up looks very similar to figure 12 on page 22, but replaces the *color selection* with a second *object selection*.

Press the **topmost** *Select Objects:* button and then click on the list element in the *Draw Window*. The button should now read as `Selected Objects: 12` or similar.

`setTip` method

Next, choose a method in the *Method:* menu. The appropriate method for setting the pointer of a list element is called *setTip*, so choose this one.

Now, also click on an arbitrary *text line* in the *Draw Window*. Notice how the editor changes to reflect that *two* objects are now selected. The *method* is automatically changed to `translate`, as this is the *only Move* method common to both *Text* and *List Element*.[3]

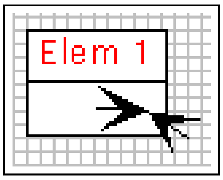
Click again on the *text line* to deselect it. If nothing happens, click on the *Select Objects:* button again to reactivate it (it must have a dark background) and again click on the selected text. Finally, change the *method* back to **setTip**.

Move Line

Now you will have to draw a *line* along which the tip is to be moved. To do so, set the *Grid* back to **5**. Select the *polyline* icon and draw a simple line as follows:

- the line's *first* point is identical to the top of the arrow,
- the line's *second* and last point – set by pressing the *middle* mouse button! – should roughly be one the same height as the starting point of the tip and lie *inside* the element box.

  An example of this line looks as follows:



Next, select this line as the *Move via* object using the **bottom** *Select Objects:* button. You can also set a *delay* and *duration* as described in section 4.12 on page 21.

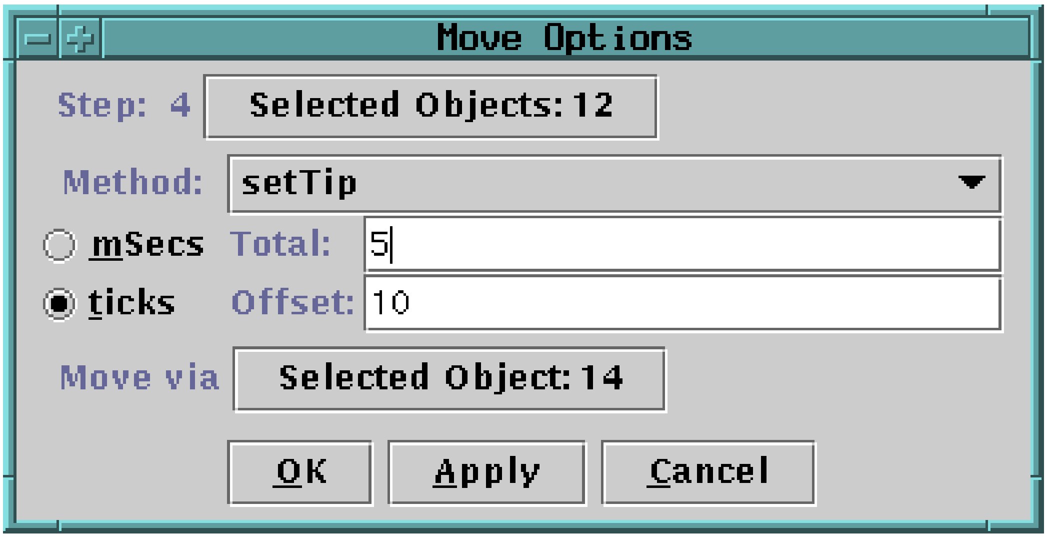
The final look of the *Move Editor* window before clicking on *OK* should resemble figure 4.13 on the following page, although your *timing* and *object numbers* may differ.

Again, use the *Run* icon to test your animation as described on on page 22 until you are satisfied with your results.

The next step is simply a repetition of previous work in which you have to do the following operations:

1. generate a new step,
2. change the color of the second text line to *blue*,
3. change the color of the third text line to *red*,
4. generate a new list element with text *Elem 2*, placed on the same *height* as the first element, but a fair distance to the right so that its left line coincides with the left line of header rectangle.

[3] ANIMAL automatically adapts the list to those methods common to *all* selected methods. If there is *no* such method, the *method* bar will read `No appropriate method!`



Step: 4 Selected Objects: 12

Method: setTip

○ mSecs Total: 5

● ticks Offset: 10

Move via Selected Object: 14

The result of these operations should resemble figure 14 on the following page. Insert another *new step* containing the following steps:

1. change color of third text line to *blue*,
2. change color of fourth text line to *red*,
3. add a *move* animator as described above.

   This time, however, draw the move line *somewhere else* at any place, such that the *second (=last)* point is *10* pixels to the *left* and *10 pixels* above the first point. Select this line as the *Move via* object and test your animation. You will see that ANIMAL uses *relative* movement – the line only shows *how* to change the object's position, and does not need to start at the targeted object.

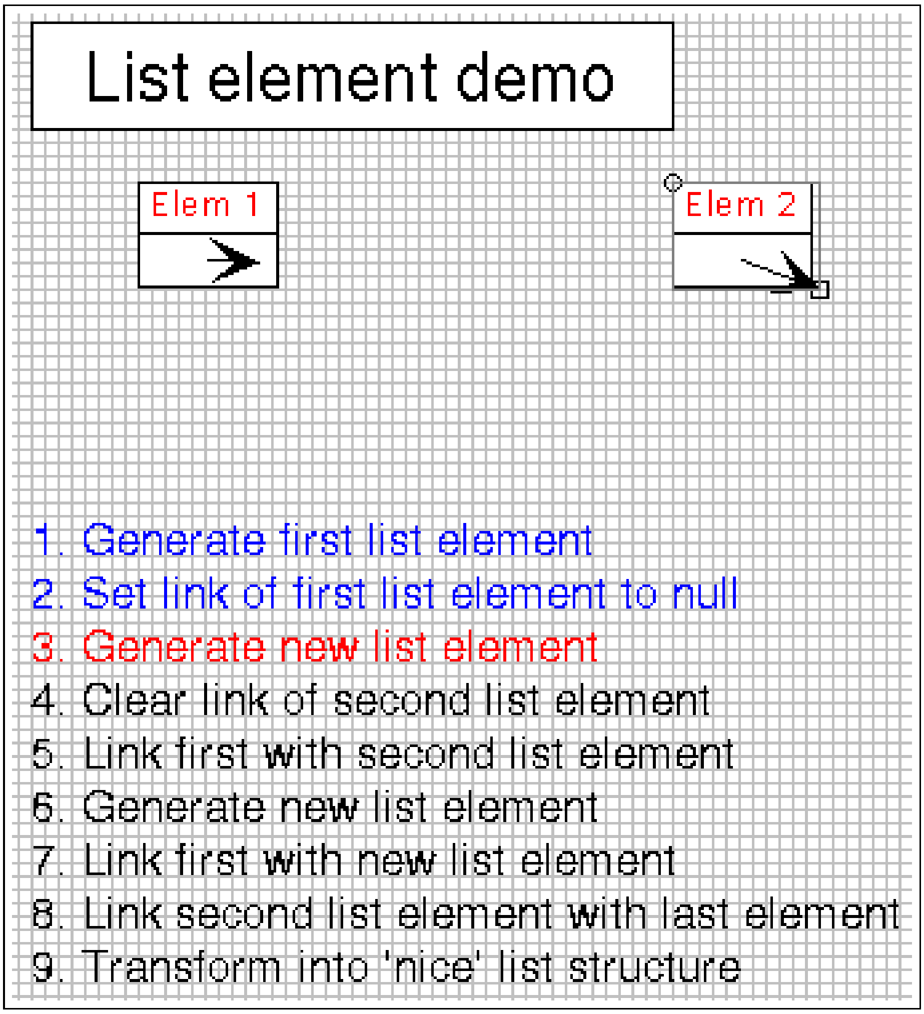
The result of these operations should resemble figure 15 on page 27. The small arrow in the display is the move line. The display of this line can be toggled using the *Show Temporary Objects* button in the *Draw Window*.

Show Temporary

Now generate a *new step* with the following operations:

1. change color of fourth text line to *blue*,
2. change color of fifth text line to *red*,
3. insert a *Move* animator for linking the two elements.

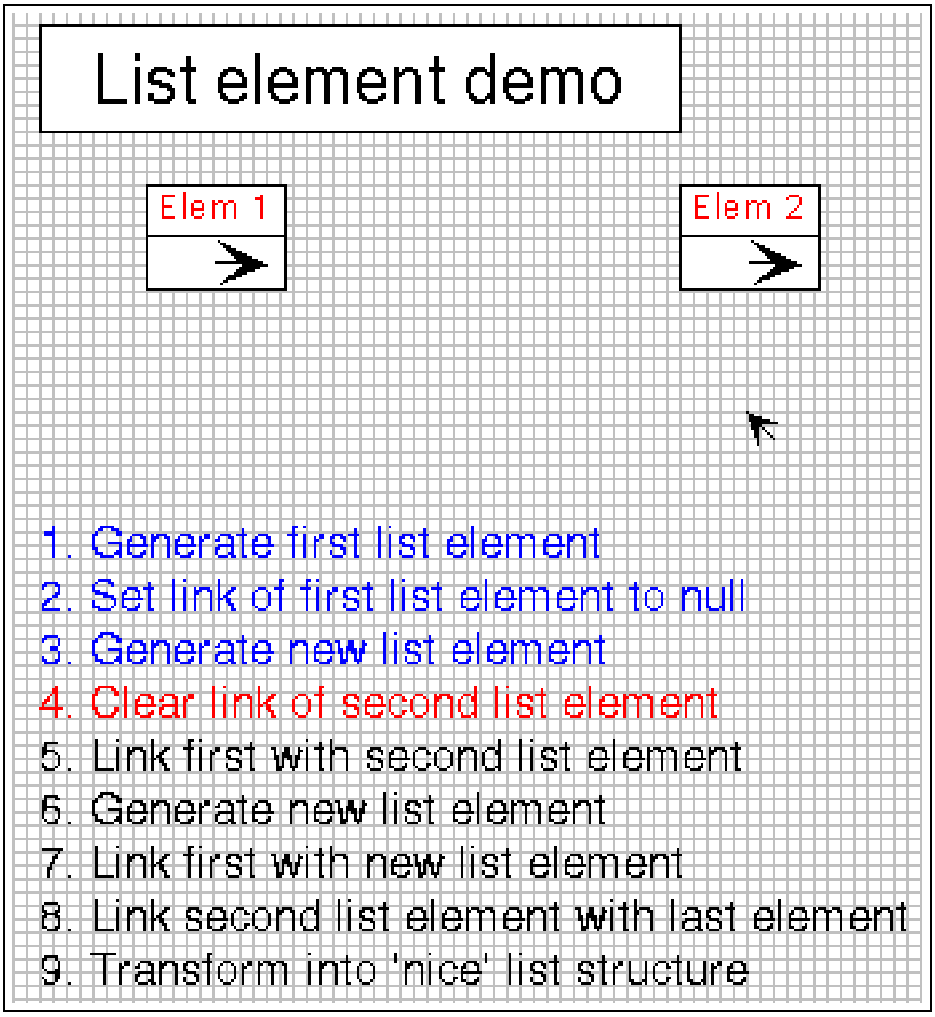
   To do so, generate a new *Move* animation effect, select the *first list element* and choose the method *setTip*.



List element demo

1. Generate first list element
2. Set link of first list element to null
3. Generate new list element
4. Clear link of second list element
5. Link first with second list element
6. Generate new list element
7. Link first with new list element
8. Link second list element with last element
9. Transform into 'nice' list structure

Figure 14: State after inserting the second element



List element demo

1. Generate first list element
2. Set link of first list element to null
3. Generate new list element
4. Clear link of second list element
5. Link first with second list element
6. Generate new list element
7. Link first with new list element
8. Link second list element with last element
9. Transform into 'nice' list structure

Figure 15: State after inserting the second element

Then click on the **bottommost** *Selected Objects:* button and draw a new polyline going from the *first list element's tip position* to the *left border* of the *second list element*. Choose appropriate timing, and test this step.

If your line is not quite the way you wanted it, **do not delete and redraw it!** Simply click on the line, and apart from the *drag points* in circle form you will notice small *squares* at both line edges[4]. Click on such a point and move the mouse with the left button pressed, and you can adapt the point to your liking.

The result of these operations should resemble figure 16. The polyline arrow in the display is the move line and was moved out of the way for better comparability.

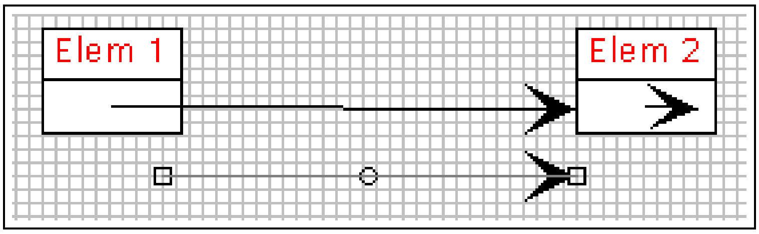


Figure 16: State after inserting the second element

The next few steps call for a repetition of the same steps. Place the *third* list element with text `Elem 3` *between* the two list elements, but *below* them.

Repeat these steps until you reach the following rough step:

4.14 Using Arc Elements

For linking the *first* and *third* list element, we will now use a *arc* component.

Begin with the usual operations, that is, adding a *new* step, changing the color of text lines *six* and *seven* and generating a new *Move* animator in which you select the *first* list element and the method `setTip`. Then, select the *Move via:* button "Select Objects:".

Arc

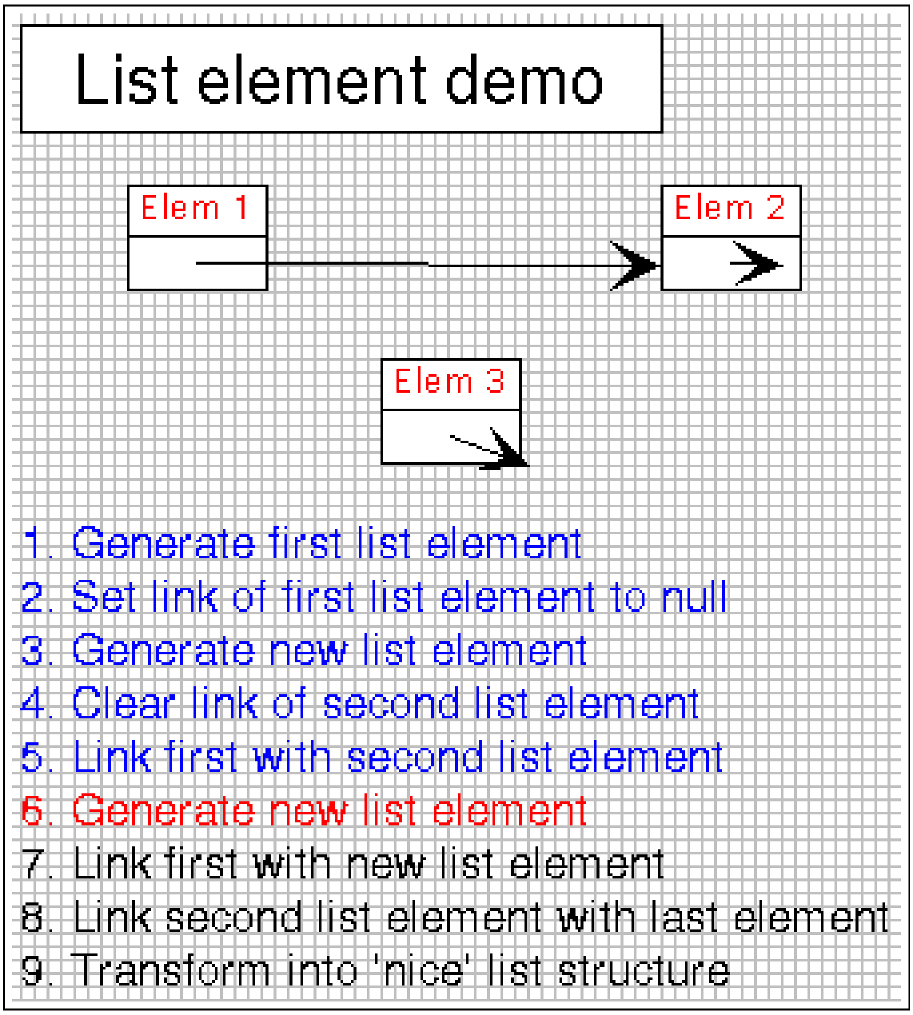
Click on the *arc* icon . Select a point *directly* next to the top right corner of the new list element as the *arc center* and click **once.**

Arc Generation

Now move the mouse to see the outline of the current arc. Try to manage that this arc line touches both the *tip* of the *first* list element and the left side of the *new list element* at the same height as that element's tip. Figure 18 on page 30 shows an example of the result. *This may take some time in trying out possible arc centers.* However, using the figure, you can determine where to place the element to make it work.

Next, click on the *first element's tip end* resting next to the second list element to mark the *arc start angle*. The next mouse click then goes to the left side of the *new* list element, and should result in something resembling figure 18 on page 30.

[4] These square are always available on *all* edges, of which this line only has two.



1. Generate first list element
2. Set link of first list element to null
3. Generate new list element
4. Clear link of second list element
5. Link first with second list element
6. Generate new list element
7. Link first with new list element
8. Link second list element with last element
9. Transform into 'nice' list structure

Figure 17: State after inserting the third element

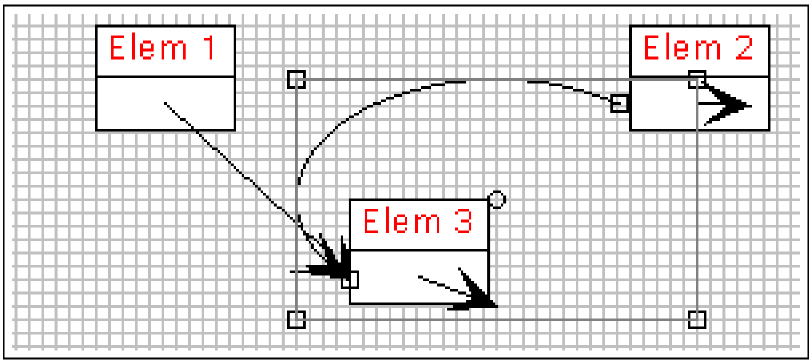


Figure 18: Linking elements using an *arc* component

Test and optimize this animation step as usual.

The next few steps are a simple of the last few actions: setting the link from the *new* element to the second list element and adapting the colors.

4.15 Diverse Options for Moving List Elements

For the last step, we want to reach a “nice” list structure in which all elements are at the same height.

To do so, you could use the *translate* method of the *Move* animation effect. However, this would also move the new element’s tip!

To avoid this problem, proceed as follows:

1. generate a new step,
2. perform the usual color highlighting on the lines 8 and 9,
3. insert a new *Move* animation effect on the *new* element, but select the method `translateWithFixedTip`. Draw a simple *polyline* starting at the *top* of the new list element and going straight up to the same height as the top of the other list elements.
4. insert a new *Move* animation effect on the *first* element, selecting the `setTip` method and using the *same* polyline as in the previous animation effect. Yes, you **can** reuse move lines – ANIMAL only forbids you to use more than a single *visible* animation on the same object. *Moving* along a line does not change the *move line*, though, so this reuse is possible.

`translateWithFixedTip`

And now. . . you’ve finished the tutorial!

If any lines show are not as straight as you want them, turn off the *Grid* and edit the lines and corresponding *move lines* until you are satisfied with the result.

List of Figures