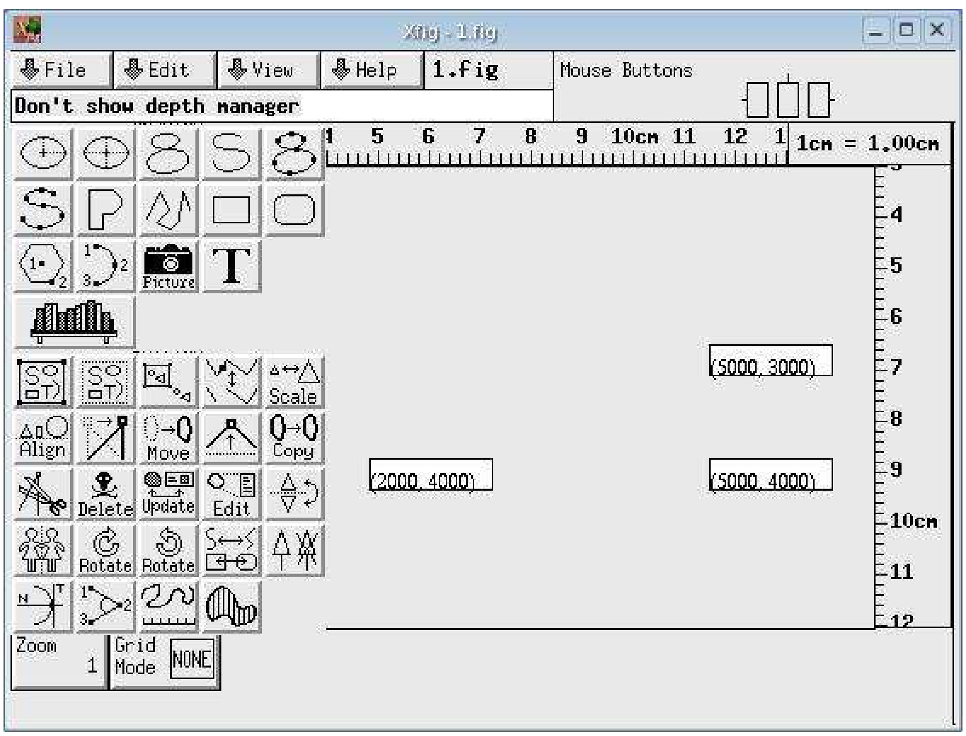
The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.

# The *display2D* operator of the relational algebra 1

## I Flat Relations

### 1. Displaying text

```
domain x intg;
domain y intg;
domain textstring strg;
relation Text(x, y, textstring) <-{
 (5000, 4000, "(5000, 4000)"),
 (2000, 4000, "(2000, 4000)"),
 (5000, 3000, "(5000, 3000)")};
NewText <- display2D( ) Text;
```

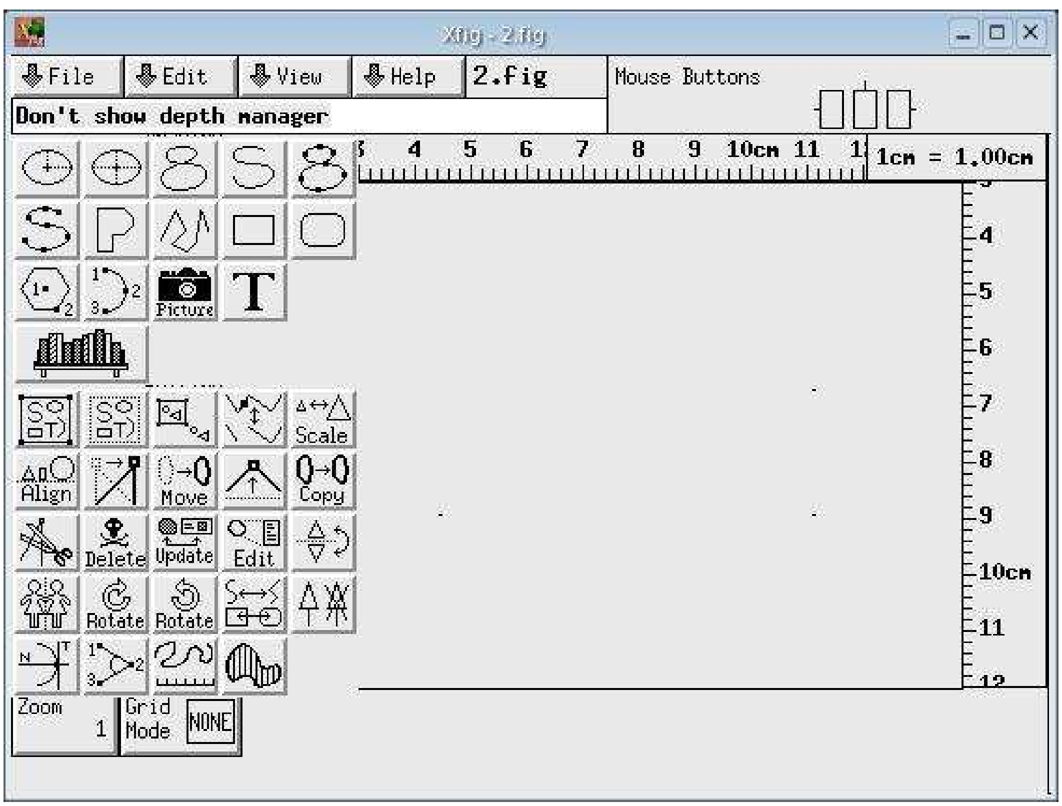


# The *display2D* operator of the relational algebra 2

## I Flat Relations

### 2. Displaying a Set of Points

```
domain x intg;
domain y intg;
relation Points (x, y) <-{
 (5000, 4000),
 (2000, 4000),
 (5000, 3000)};
NewPoints <- display2D( ) Points;
```

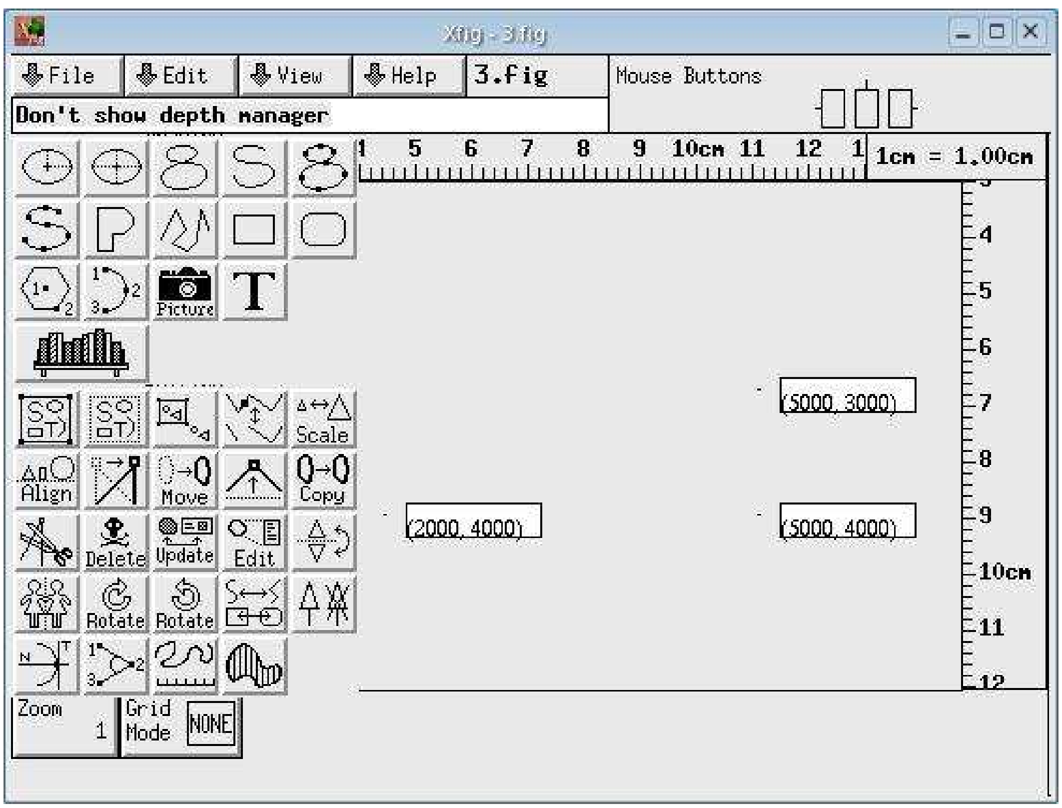


# The *display2D* operator of the relational algebra 3

## I Flat Relations

### 3. Displaying a Set of Labelled Points

```
domain label strg;
domain lc intg;
relation LabelledPoints (x, y, lc, label) <-{
 (5000, 4000, 0, "(5000,4000)"),
 (2000, 4000, 0, "(2000, 4000)"),
 (5000, 3000, 0, "(5000, 3000)")};
NewLabelledPoints <- display2D( ) LabelledPoints;
```

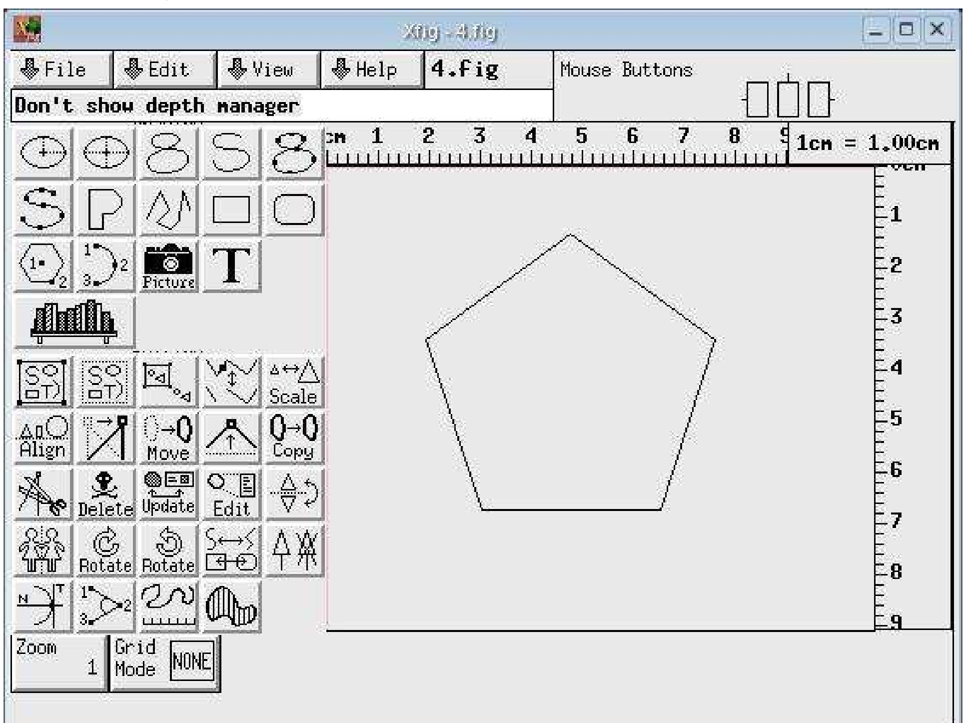


# The *display2D* operator of the relational algebra 4

## I Flat Relations

### 4. Displaying a Set of Lines

```
domain x1 intg;
domain y1 intg;
domain x2 intg;
domain y2 intg;
relation Lines(x1, y1, x2, y2) <- {
 (1363, 3013, 2942, 3010),
 (2942, 3010, 3426, 1508),
 (3426, 1508, 2148, 583),
 (2148, 583, 873, 1514),
 (873,1514, 1363, 3013)};
NewLines <- display2D( ) Lines;
```



# The *display2D* operator of the relational algebra 5

## I Flat Relations

4. Displaying a Set of Lines, Part 2

A graph is a set of lines:

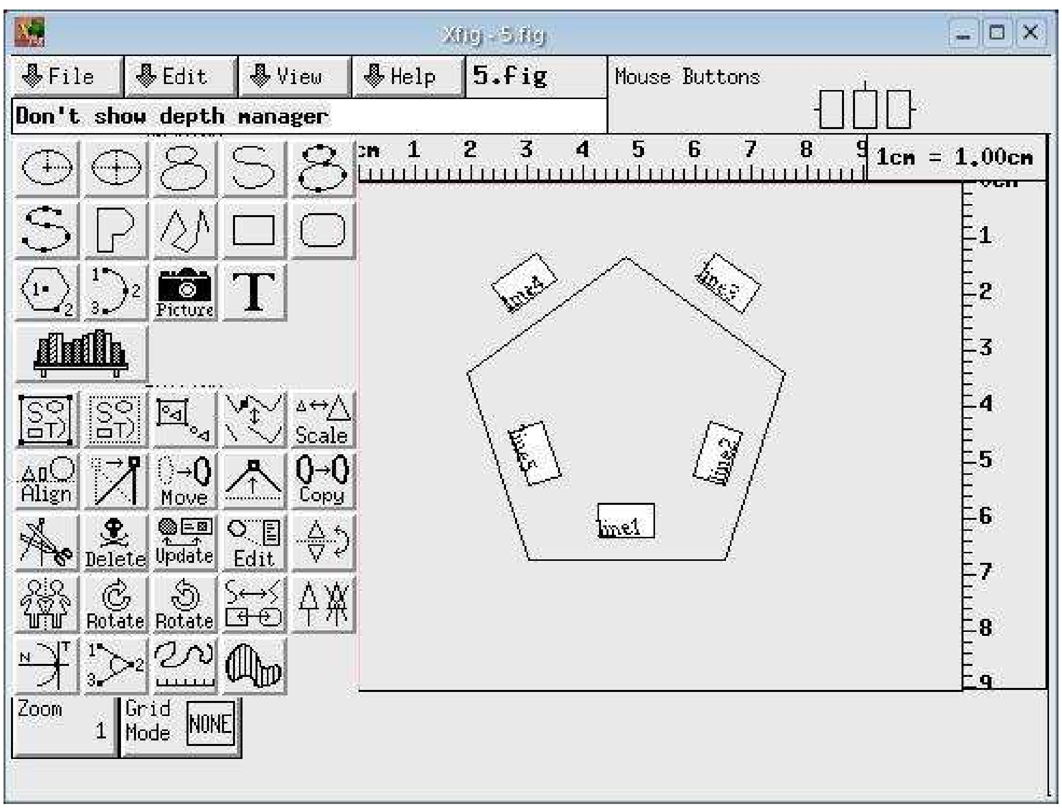
```
relation f(x,y) <- {(-2.0,-5.0),(-1.5,-2.5),
 (-1.0, 0.0),(-0.5, 0.625), (0.0, 1.0),
 (0.5, 1.875),(1.0, 4.0),(1.5, 6.875),(2.0,17.0)};
let x1 be fun succ of x order x;
let y1 be fun succ of y order x;
newf <- display2D() where x1>x in f;
```

# The *display2D* operator of the relational algebra 6

## I Flat Relations

### 5. Displaying a Set of Labelled Lines

```
relation LabelledLines(x1, y1, x2, y2, lc, label) <-{
 (1363, 3013, 2942, 3010, 0,"line1"),
 (2942, 3010, 3426, 1508, 0, "line2"),
 (3426, 1508, 2148, 583, 0, "line3"),
 (2148, 583, 873, 1514, 0, "line4"),
 (873,1514,1363,3013, 0, "line5")};
NewLabelledLines <- display2D( ) LabelledLines;
```

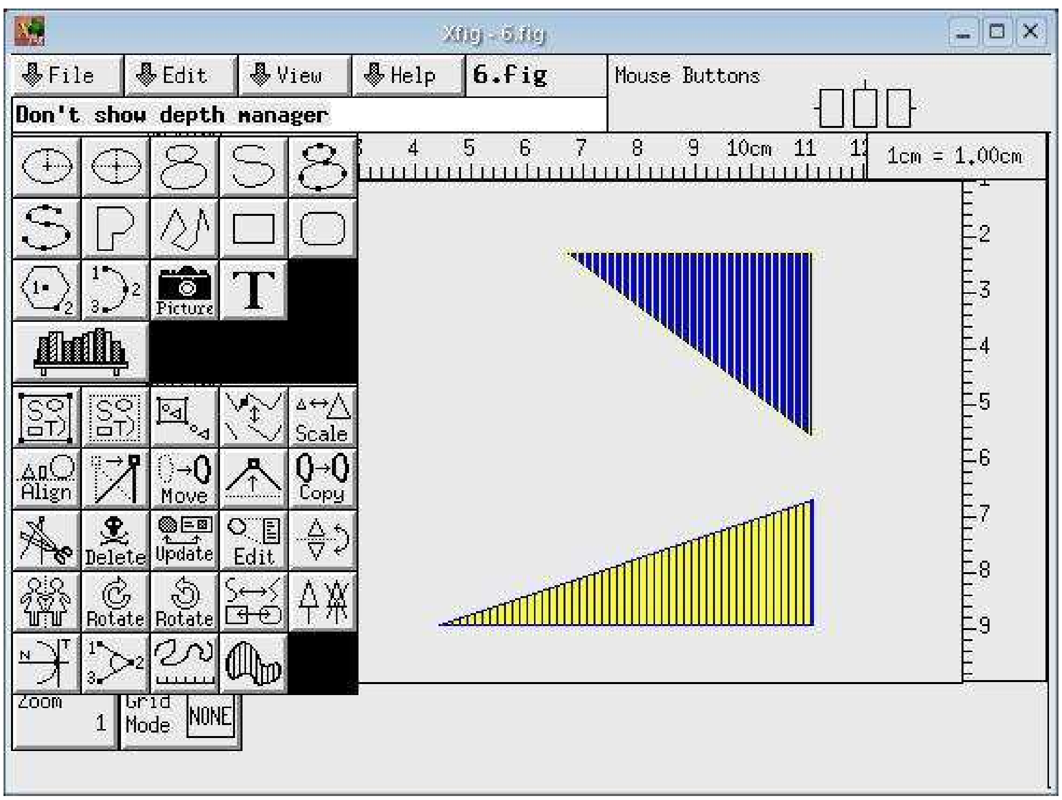


# The *display2D* operator of the relational algebra 7

## I Flat Relations

### 6. Displaying a Set of Triangles

```
domain x1, y1, x2, y2, x3, y3, lc, fc, fp intg;
relation Triangle(x1, y1, x2, y2, x3, y3, lc, fc, fp)<-{
 (5000, 4000, 2000, 4000, 5000, 3000, 1, 6, 50),
         (3000, 1000, 5000, 1000, 5000, 2500, 6, 1, 50)};
NewTriangle <- display2D( ) Triangle;
```

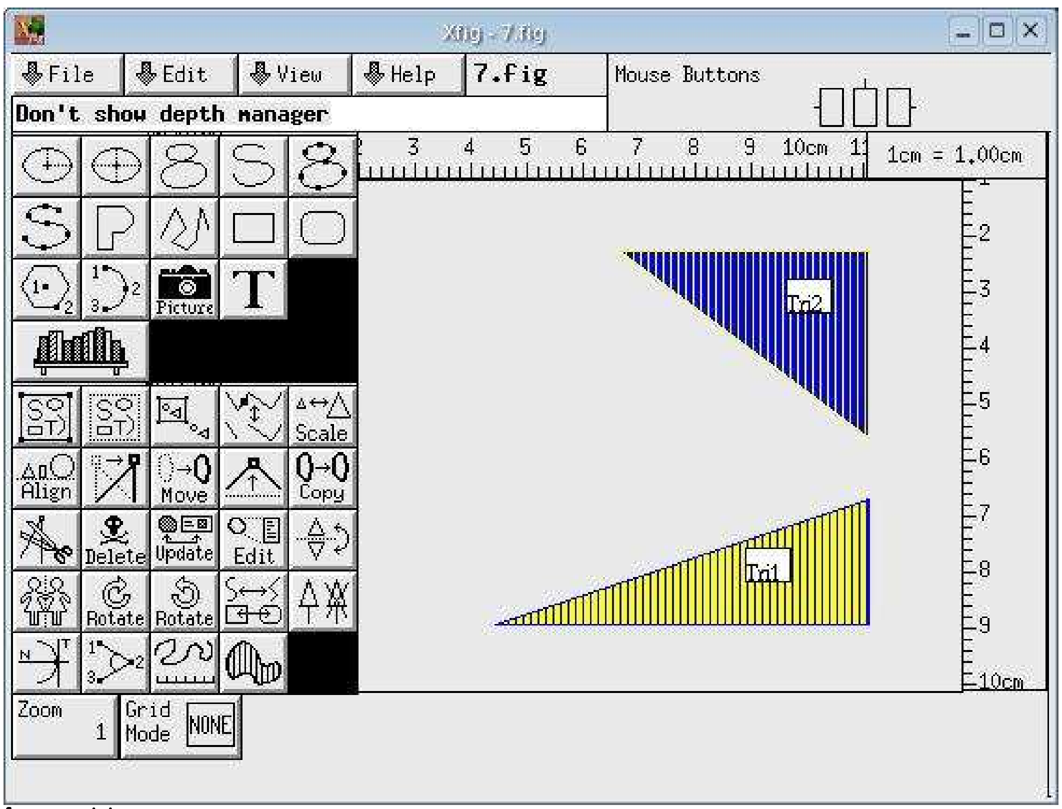


# The *display2D* operator of the relational algebra 8

## I Flat Relations

### 7. Displaying a Set of Labelled Triangles

```
relation LabelledTriangle
(x1, y1, x2, y2, x3, y3, lc, fc, fp, label)<- {
 (5000, 4000, 2000, 4000, 5000, 3000, 1, 6, 50, "Tri1"),
 (3000, 1000, 5000, 1000, 5000, 2500, 6, 1, 50, "Tri2")};
NewLabelledTriangle <- display2D( ) LabelledTriangle;
```

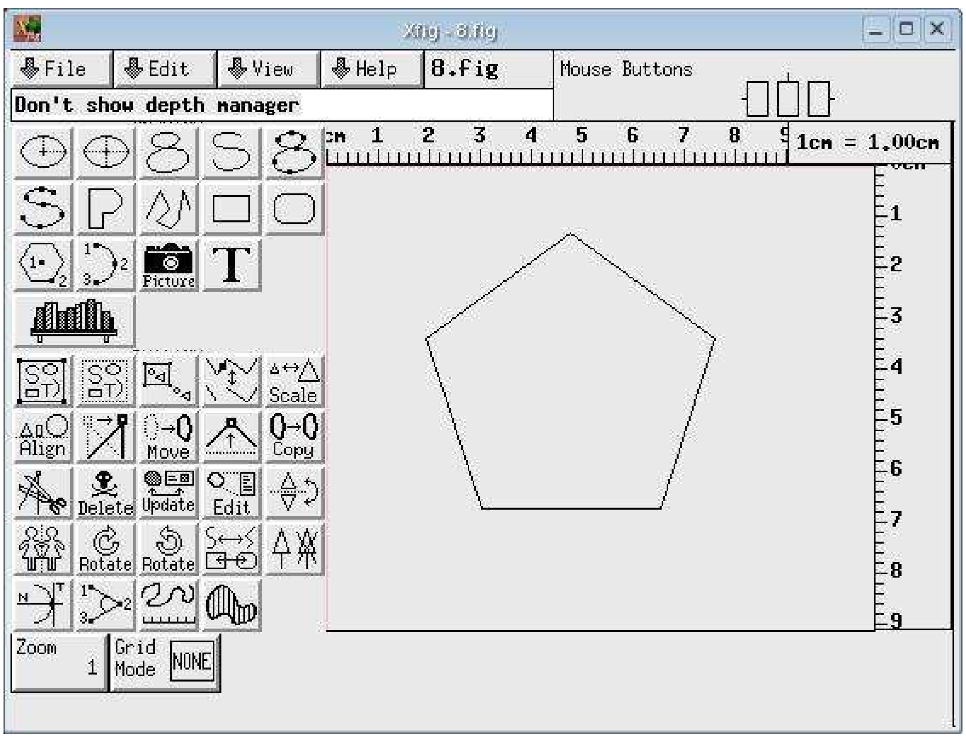


# The *display2D* operator of the relational algebra 9

## I Flat Relations

### 8. Displaying a Sequenced Polyline

```
relation Polyline(x, y, sq) <-{
 (1363, 3013, 1),
 (2942, 3010, 2),
 (3426, 1508, 3),
 (2148, 583, 4),
 (873, 1514, 5),
 (1363, 3013, 6)};
NewPolyline <- display2D( ) Polyline;
```

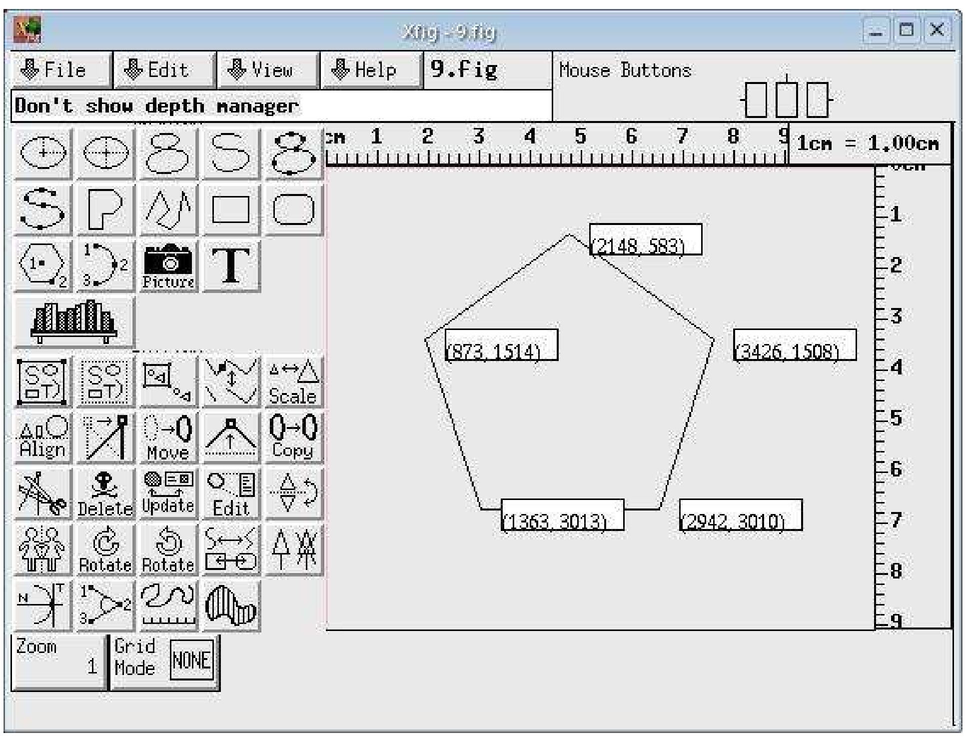


# The *display2D* operator of the relational algebra 10

## I Flat Relations

### 9. Displaying a Sequenced Polyline with Labelled Vertices

```
relation LabelledVertexPolyline(x, y, sq, lc, label) <-{
 (1363, 3013, 1, 0, "(1363, 3013)"),
 (2942, 3010, 2, 0, "(2942, 3010)"),
 (3426, 1508, 3, 0, "(3426, 1508)"),
 (2148, 583, 4, 0, "(2148, 583)"),
 (873, 1514, 5, 0, "(873, 1514)"),
 (1363, 3013, 6, 0, "(1363, 3013)")};
NewLabelledVertexPolyline <- display2D( ) LabelledVertexPolyline;
```

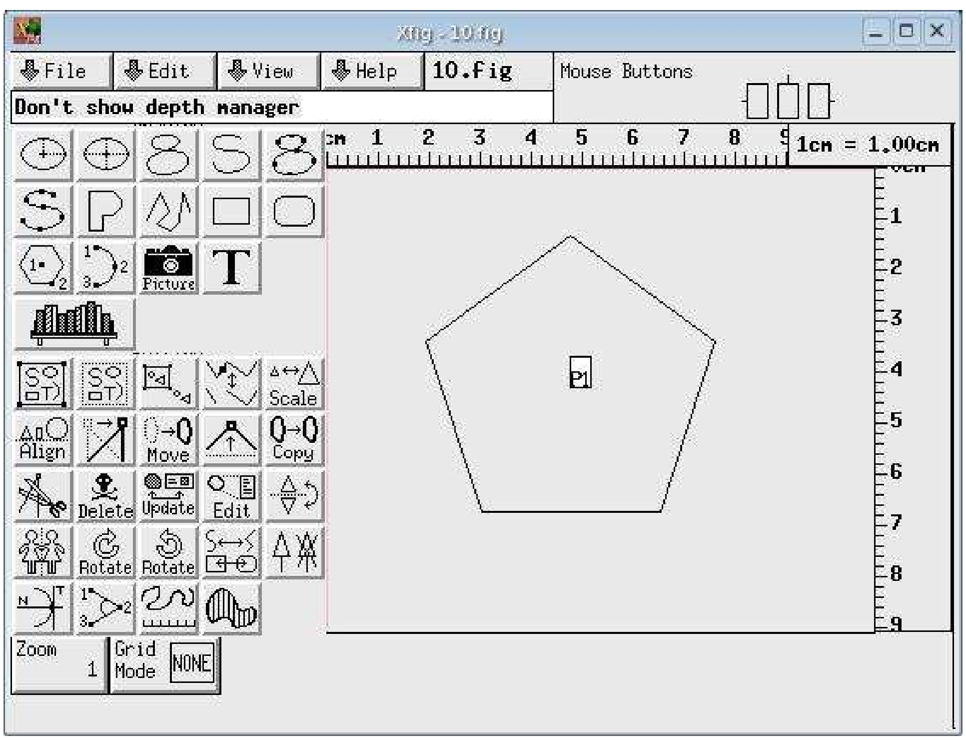


# The *display2D* operator of the relational algebra 11

## II Nested Relations

### 1. Displaying a Sequenced Polyline with a Label

```
domain lc intg;
domain Polyline (x, y, sq);
relation NestedPolyline ( label, lc, Polyline)<- {
 ("P1", 0, {(1363, 3013, 1), (2942, 3010, 2),
    (3426, 1508, 3), (2148, 583, 4),
    (873, 1514, 5), (1363, 3013, 6)})};
NewNestedPolyline <- display2D( ) NestedPolyline;
```

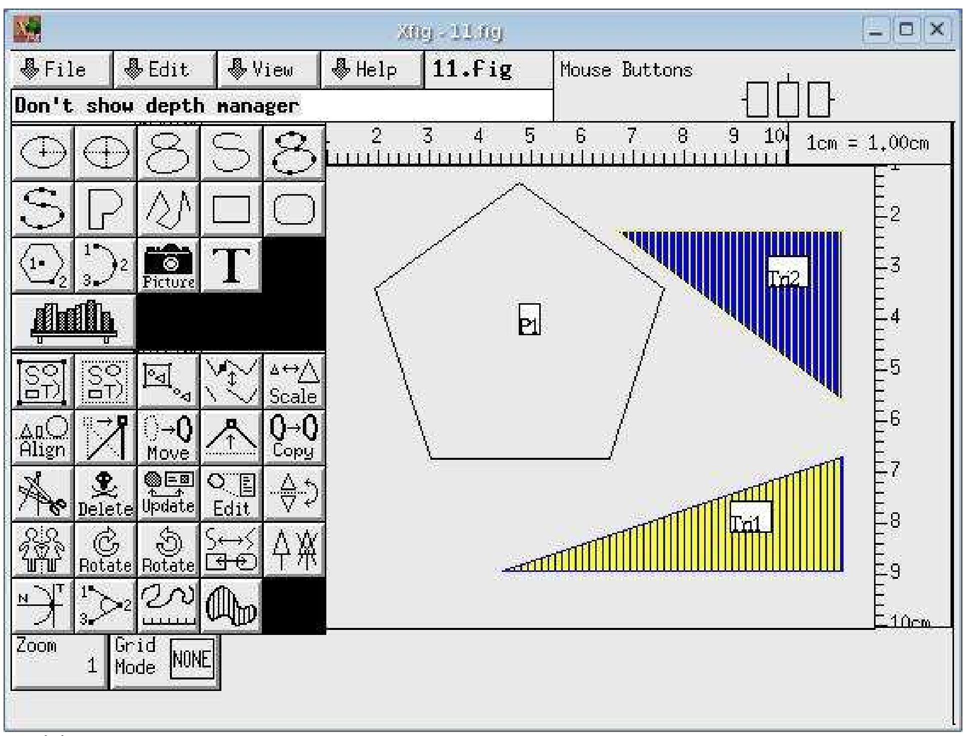


# The *display2D* operator of the relational algebra 12

## II Nested Relations

### 2. Displaying Several Polylines or a Combination of Different Shapes

```
domain Polyline (x, y, sq);
domain NestedPolyline (label, lc, Polyline);
domain LabelledTriangle (x1, y1, x2, y2, x3, y3, lc, fc, fp, label);
relation Graph (NestedPolyline, LabelledTriangle)<-{
 ({("P1", 0, {(1363, 3013, 1), (2942, 3010, 2), (3426, 1508, 3),
      (2148, 583, 4), (873, 1514, 5), (1363, 3013, 6)})},
 {(5000, 4000, 2000, 4000, 5000, 3000, 1, 6, 50, "Tri1"),
  (3000, 1000, 5000, 1000, 5000, 2500, 6, 1, 50, "Tri2")})};
NewGraph<-display2D( ) Graph;
```



# The *display2D* operator of the relational algebra 13

## III Vocabulary

```
pr .vocabulary;
+----------------------+----------------------+
| .attribute    | .meaning        |
+----------------------+----------------------+
| x             | cart1           |
| x1            | cart1           |
| x2            | cart1           |
| x3            | cart1           |
| x4            | cart1           |
| y             | cart2           |
| y1            | cart2           |
| y2            | cart2           |
| y3            | cart2           |
| y4            | cart2           |
| sq            | sequence        |
| lc            | line_colour     |
| fc            | fill_colour     |
| tc            | text_colour     |
| fp            | fill_pattern    |
| ls            | line_style      |
| lt            | line_thickness  |
| dl            | dash_length     |
| ft            | font            |
| fs            | font_size       |
| dp            | depth           |
| js            | join_style      |
| cs            | cap_style       |
| fa            | forward_arrow   |
| ba            | backward_arrow  |
+----------------------+----------------------+
relation .vocabulary has 25 tuples
```

# The *display2D* operator of the relational algebra 14

## III Vocabulary

```
relation TextVocabulary(.attribute,.meaning) <-
 {(a, cart1), (b, cart2)};
domain a intg;
domain b intg;
domain textstring strg;
relation Text2(a,b,textstring) <-
 {(5000, 4000, "(5000,4000)"),
  (2000, 4000, "(2000,4000)"),
  (5000, 3000, "(5000,3000)")};
NewText2 <- display2D(TextVocabulary) Text2;
```

(Same result as slide "1. Displaying text")

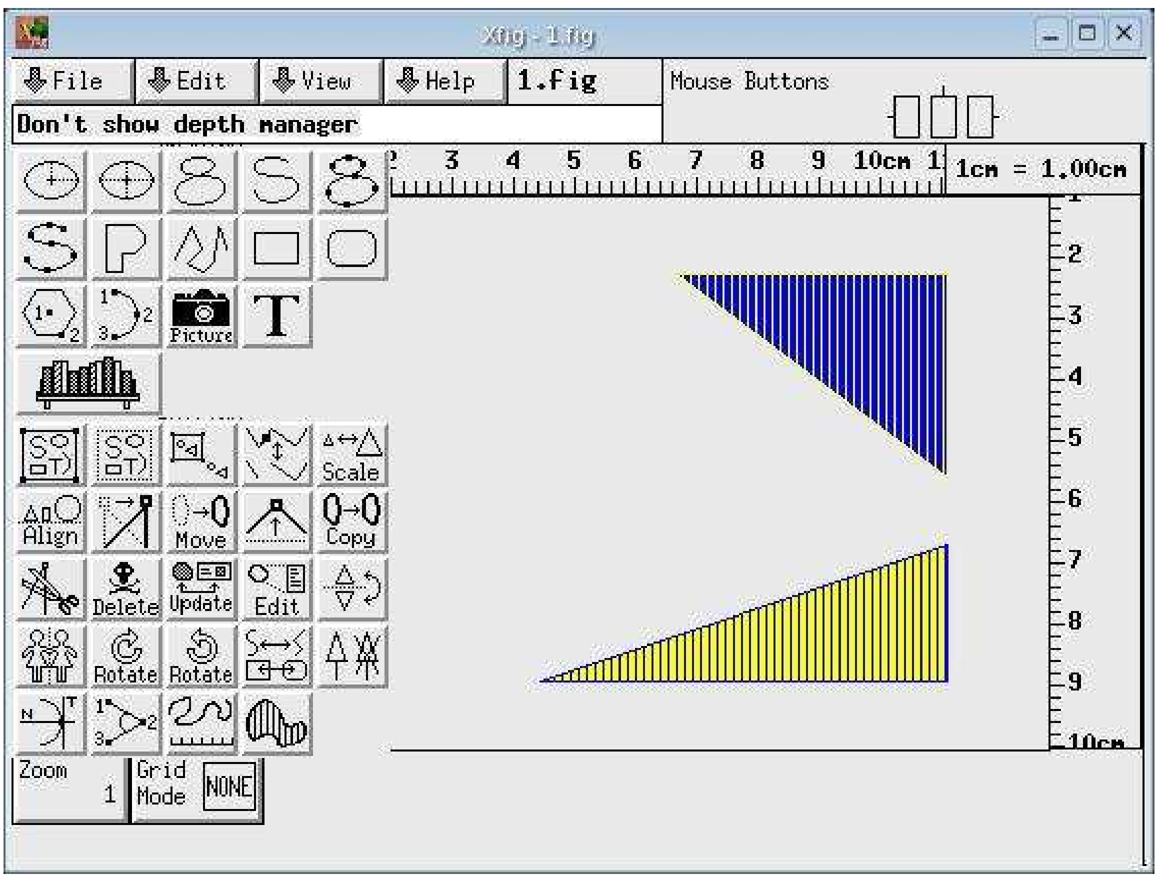
 

# The *display2D* operator of the relational algebra 15

## IV Updating via the display

E.g., using the two-triangle example we drew earlier:

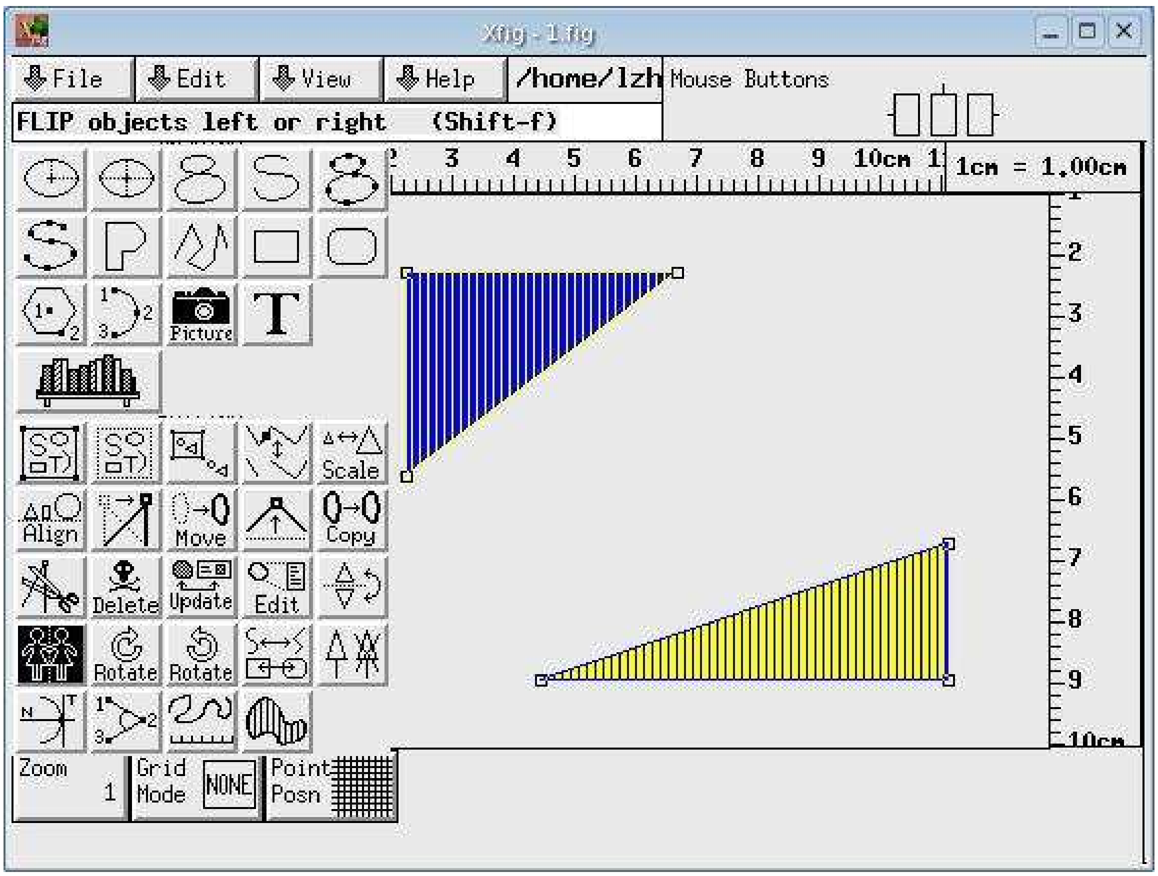
```
NewTriangle <- display2D( ) Triangle;
```



# The *display2D* operator of the relational algebra 16

## IV Updating via the display

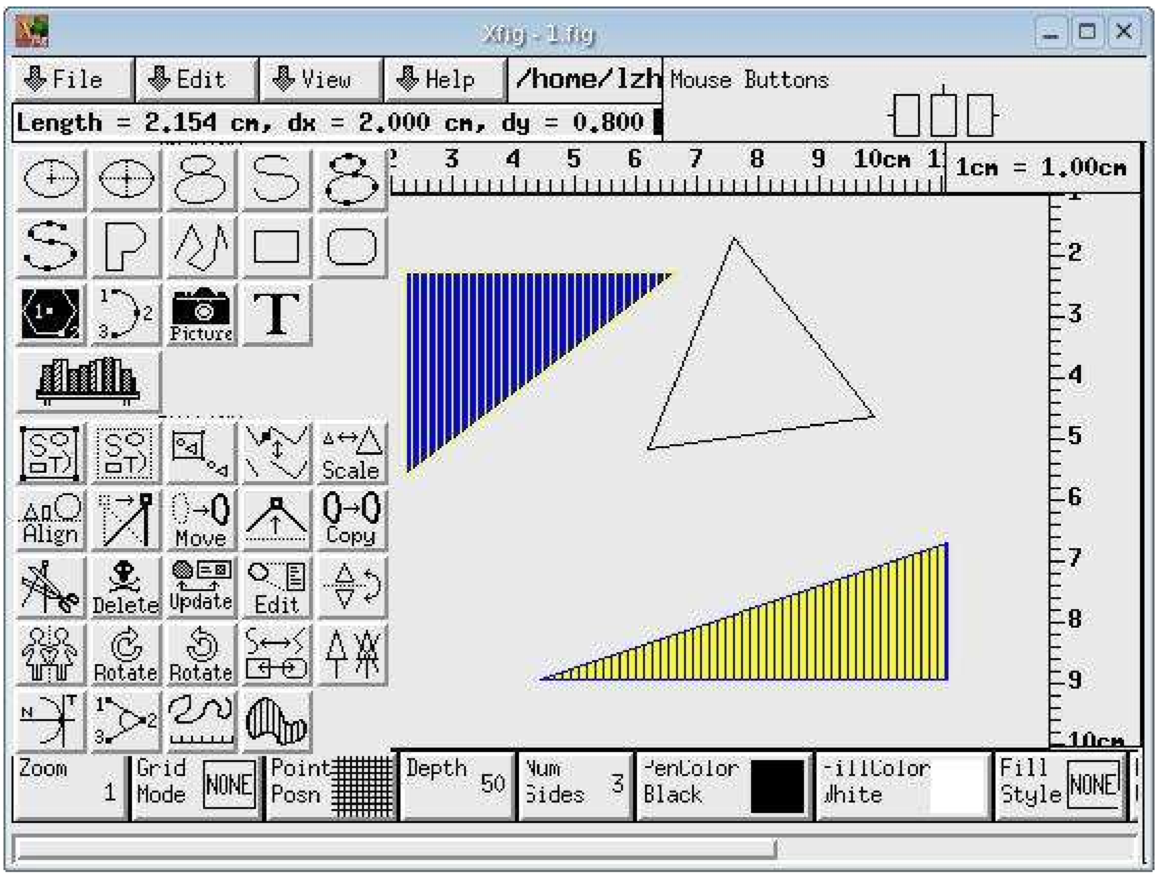
1. Flip the top triangle using the Xfig toolbar;



# The *display2D* operator of the relational algebra 17

## IV Updating via the display

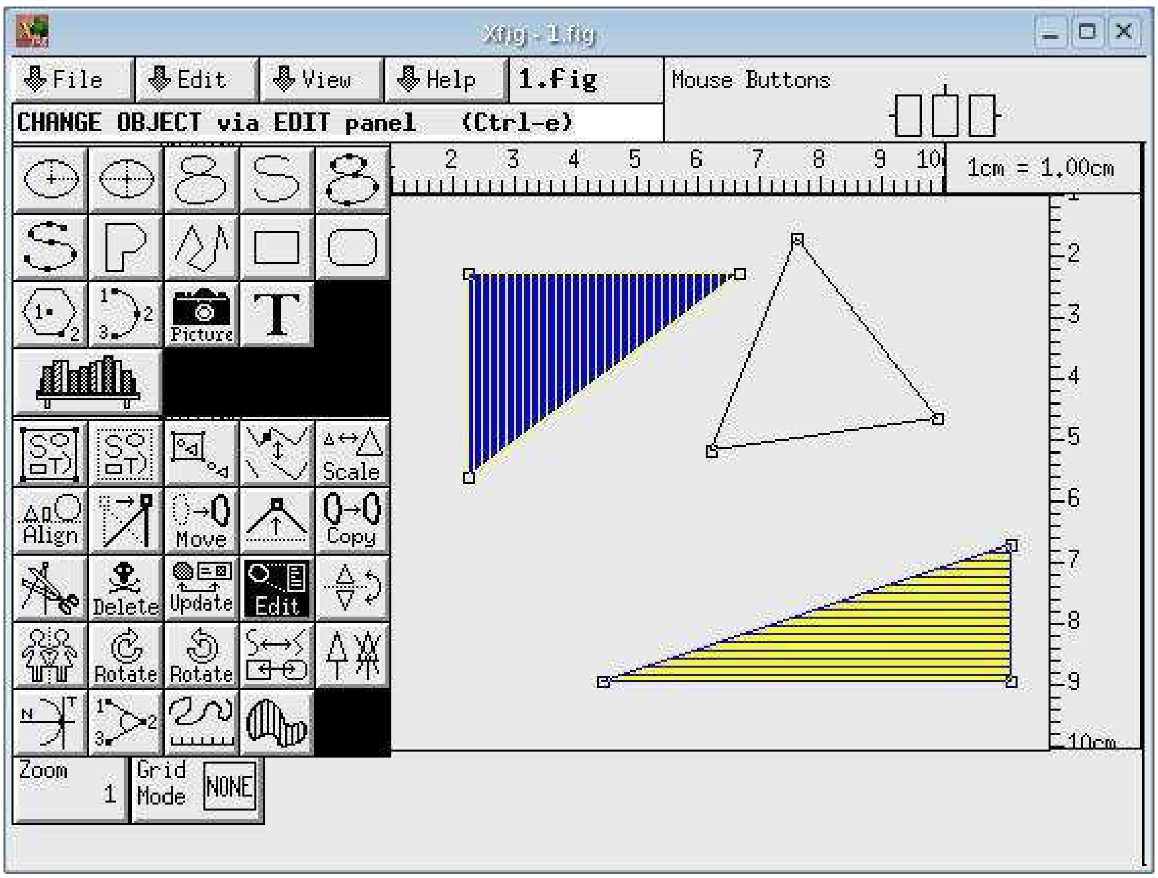
2. Draw a third triangle using Xfig tools;



# The *display2D* operator of the relational algebra 18

## IV Updating via the display

3. Edit the bottom triangle, using Xfig tools, to fill with horizontal lines instead of vertical.



# The *display2D* operator of the relational algebra 19

## IV Updating via the display

```
pr NewTriangle;
+----------------------+----------------------+--------------+
|  x1  |  y1  |  x2  |  y2  |  x3  |  y3  |  lc  | fc  |  fp |
+----------------------+----------------------+--------------+
| 5000 | 4000 | 2000 | 4000 | 5000 | 3000 |   1  |   6 |  49 |
| 3000 | 1000 | 1000 | 1000 | 1000 | 2500 |   6  |   1 |  50 |
| 4455 | 2070 | 3417 |  751 | 2793 | 2309 |   0  |   7 |  -1 |
+----------------------+----------------------+--------------+
relation NewTriangle has 3 tuples
```

Compare the original version:

```
pr Triangle;
+----------------------+----------------------+--------------+
|  x1  |  y1  |  x2  |  y2  |  x3  |  y3  |  lc  | fc  |  fp |
+----------------------+----------------------+--------------+
| 5000 | 4000 | 2000 | 4000 | 5000 | 3000 |   1  |  6  |  50 |
| 3000 | 1000 | 5000 | 1000 | 5000 | 2500 |   6  |  1  |  50 |
+----------------------+----------------------+--------------+
relation Triangle has 2 tuples
```

# The *display2D* operator of the relational algebra 20

## IV Updating via the display

!! Updates which add new attributes are not allowed (this includes introducing non-default values where defaults were originally used).

!! Updates which need to change a flat relation to a nested relation are not allowed.

!! Updates to nested relations are not implemented.

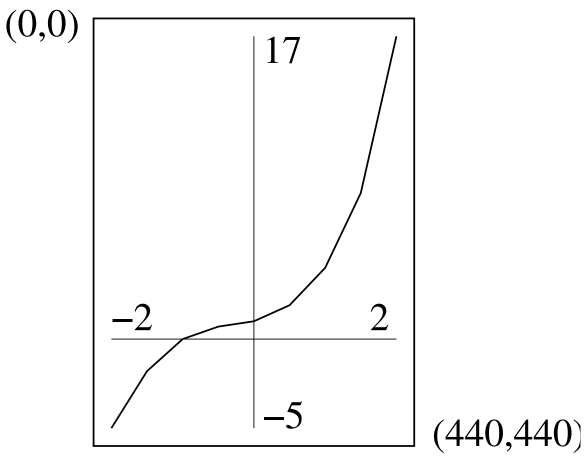
Ref. Lili Zhu, "A Generalized Two-Dimensional Display Editor for Relations", M.Sc. Thesis, SOCS, Dec. 2005.
www.cs.mcgill.ca/~tim/cv/theses/zhuLiliThesis.pdf

# The *display2D* operator of the relational algebra 21

## V Application: scaling to a box

$$x^3 + x^2 + x + 1$$

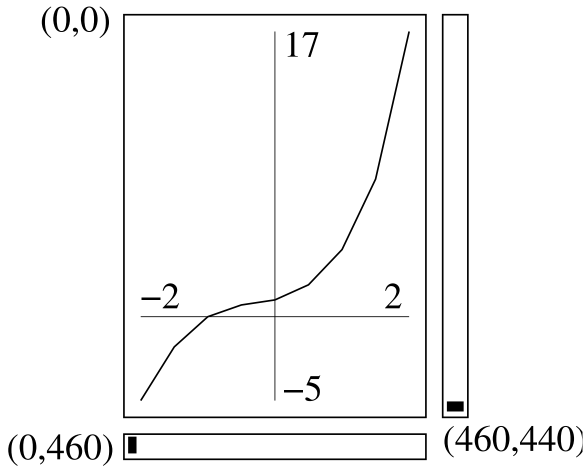



| $f(x$ | $y)$ | *box* | | |
|---|---|---|---|---|
| -2 | -5 | (*seq* | $X$ | $Y$) |
| -1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 2 | 0 | 440 |
| 1 | 4 | 3 | 440 | 440 |
| 2 | 17 | 4 | 440 | 0 |

**let** *xmin* **be red min of** $x$; -2
**let** *Xmin* **be red min of** $X$; 0
**let** *xspan* **be** (**red max of** $x$)− *xmin*; 4
**let** *Xspan* **be** (**red max of** $X$)− *Xmin*; 440
**let** *xscale* **be** *Xspan*/*xspan*; 110
**let** *xdispl* **be** *Xmin* − *xmin***xscale*; 220
**let** *X* **be** *xscale**$x$ + *xdispl*;
<< sim. $Y$ >>

# The *display2D* operator of the relational algebra 22

## V Unimplemented: adding sliders




*xslider*

| (*xloc* | *yloc* | *len* | *rangemin* | *rangemax* | *xzoom*) |
|---|---|---|---|---|---|
| 0 | 460 | 440 | 1 | 100 | 1 |

*yslider*

| (*xloc* | *yloc* | *len* | *rangemin* | *rangemax* | *yzoom*) |
|---|---|---|---|---|---|
| 460 | 440 | 440 | 1 | 100 | 1 |

# The *display2D* operator of the relational algebra 23

## V Unimplemented: sliders and zooming

| *plot* | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ( $f$ | | *box* | | | *xslider* | | *yslider* | | ) |
| ( $x$ | $y$ ) | (*seq* | $X$ | $Y$ ) | (.. | *xzoom*) | (.. | *yzoom*) | |
| -2 | -5 | 1 | 0 | 0 | .. | 1 | .. | 1 | |
| -1 | 0 | 2 | 0 | 440 | | | | | |
| 0 | 1 | 3 | 440 | 440 | | | | | |
| 1 | 4 | 4 | 440 | 0 | | | | | |
| 2 | 17 | | | | | | | | |

**comp post:change:***plot*/*xslider*[*xzoom*]() **is**
{ **let** *xmin* **be** (**red min of** $x$)/[*xzoom*] **in** *xslider*;
  **let** *Xmin* **be** [**red min of** $x$] **in** *box*;
  **let** *xspan* **be** (**red max of** $x$)/([*xzoom*] **in** *xslider*) − *xmin*;
  **let** *Xspan* **be** ([**red max of** $X$] **in** *box*) − *Xmin*;
  **let** *xscale* **be** *Xspan*/*xspan*;
  **let** *xdispl* **be** *Xmin* − *xmin***xscale*;
  **let** *X* **be** *xscale**$x$ + *xdispl*;
  **let** *X1* **be fun succ of** $X$ **order** $X$;
  << sim. $Y$ >>
  **let** *F* **be** [*X*,*Y*,*X1*,*Y1*] **where** $X <$ *X1* **in** $f$;
  **let** *Box* **be** [*X*,*Y*,*X1*,*Y1*] **in** *box*;

  **display2D**(*vocab*)[*F*,*Box*,*xslider*,*yslider*] **in** *plot*;

}

# The *display2D* operator of the relational algebra 24

## V Unimplemented: sliders and zooming

where

| *vocab* | |
|---|---|
| (*.attribute* | *.meaning*) |
| `x` | `cart1` |
| `y` | `cart2` |
| `xslider` | `slider1` |
| `yslider` | `slider2` |
| `X` | `cart1` |
| `Y` | `cart2` |
| `X1` | `cart1` |
| `Y1` | `cart2` |

We could go on to frame all this in a window with controls for moving and resizing.