Inception document for the Prezoom project.

Theodore S Norvell
Electrical and Computer Engineering
Memorial University

2020 October

Abstract
A short outline of the requirements for the Prezoom system.

1 Need and purpose

Presentation software (such as PowerPoint, Keynote, and Impress) generally has a slide-by-slide nature that is choppy oriented toward static images and text.

Some attempts to create a more fluid style of presentations have been made, see Prezi for example. But Prezi is still geared toward static images and text with little support for animation; movement is confined to "camera moves" such as panning, zooming, and tilting.

PowerPoint and its imitators allow some animation within a slide, but the editing of animations can be difficult and awkward and scales poorly — making the animation twice as complex requires much more than twice the work.

The goal of this project is to create a new form of presentation software that will be rooted in drawing and animation.

2 Proposed approach

2.1 State

At heart the new system should be a drawing program similar to Visio or Pencil Project. It should be possible to place 2-dimensional, graphical objects such as rectangles, ovals, lines, and text areas on an infinite (or very large) background sheet. These graphical objects should have various attributes such as outline colour, fill colour, position, horizontal size, vertical size, depth,[1] orientation, text content, visibility, etc. (Different kinds of graphical

[1] The depth is sometimes called the z-index. It is used to control the order in which objects are drawn to the screen. Objects with larger depth numbers are drawn first, this ensures that they appear to be 'behind' overlapping objects that are drawn later.

objects might require different sets of attributes. For example a line might have an attribute for specifying what sort of arrow head or tail it has, whereas that attribute does not make sense for an oval.) The kind itself might even be considered to be an attribute; e.g., the difference between a square and an oval might simply be the value of an attribute.

2.2 Timeline, transitions, and presentation mode

In order make presentations, the user should be able to move their view back and forth along a "time line" and to edit that "time line". A time line can be thought of as an alternating sequence of one or more "states" (or "key-frames") and one fewer "transitions", i.e.,

$$s_0 \; t_0 \; s_1 \; t_1 \; \cdots \; s_i \; t_i \; s_{i+1} \; \cdots \; s_{n-1} \; t_{n-1} \; s_n$$

where $n \geq 0$. In each state, each attribute of each graphical object has a specific value. In the next state objects might have different values for some of their attributes. Transitions are associated with a trigger event, a length of time, and a list of changes to the state. Trigger events could be a press of a key, the passage of time, and maybe some others.

In "presentation mode" the system should start in state s_0. When a trigger event for t_0 happens, all the changes to the state which are associated with the transition t_0 start to happen; these changes should be complete after the length of time associated with t_0, after that amount of time the state displayed should be s_1. Then the process starts again with t_1 and so on until the state is s_n. When the time associated with the transition is greater than 0, the changes to the attribute values should be gradual. For example, if t_i takes 1s and an object's position changes from $(0, 0)$ in state s_i to $(300, 600)$ in state s_{i+1},then (assuming a 30 frames per second refresh rate), the object might be shown at each of these locations

$$(0, 0), (10, 20), (20, 40), \ldots, (290, 580)$$

for 1/30th of a second before settling at location $(300, 600)$.[2]

[2]This example assumed a linear interpolation, but other kinds of interpolation can be used and are often desirable. Linear interpolation can be represented by

$$v_t = \frac{t - t_a}{t_b - t_a} \times (v_b - v_a) + v_a$$

An "ease-in ease-out" interpolation can be represented by

$$v_t = \begin{cases} v_a & \text{, when } t = t_a \\ \sigma\left(\frac{t-t_a}{t_b-t_a}\right) \times (v_b - v_a) + v_a & \text{, when } t_a < t < t_b \\ v_b & \text{, when } t = t_b \end{cases}$$

where $\sigma(x)$ is some sigmoid function such as

$$\sigma(x) = \frac{\tanh((x - 0.5) \times 5) + 1}{2}$$

For 2-D attributes such as position, the trajectory could by curved in space as well as time.

2.3 Camera and camera movement

Another aspect of the state is the camera. The camera can be thought of as rectangular object that has a position, a horizontal size, a vertical size,[3] and an orientation. In presentation mode, the portion of the background sheet that corresponds to the camera should be portion shown to the user. So, if the camera moves to the left in a transition, what the viewer of the presentation will see is the graphical objects moving to the right.

Note: the way I've described the time-line as an alternating sequence of states and transitions might not be the best way to represent it within the computer. This is just my way of explaining the idea. Furthermore, it's not really clear how best to present the timeline to the user during editing: should they see a visualization of the sequence of states,

2.4 Editing mode

As I said above, when in editing mode, it should be possible to modify the timeline in various ways, e.g. by inserting new transitions and states, by deleting one or more states and transitions, etc.

And it should be possible to modify states – e.g. by changing the values of attribute of objects.

And it should be possible to modify transitions, e.g., by modifying the trigger event, the time it takes, and what sort of interpolation should be used.

2.5 Features

Here is a list of desired features. The features not marked with a (*) define the "minimal viable product".

Features shown with a (*) should be considered "nice to have"; it's unlikely that all will be implemented in the time frame of the course, but some might; in any case they should be considered as likely future changes that your design should accommodate. For example your design should easily accommodate adding new kinds of graphical objects and new attributes. And even if it doesn't support undo and redo, you should design the system in a way that will make it easy to add this feature in the future.

- (Save) It should be possible to save a presentation to file (or to a cloud server).
- (Load) It should be possible to read a presentation from a file (or from a cloud server).
- A presentation should consist of at least one sheet, a sequence of one or more states, separated by transitions, and zero or more graphical objects.
- (Edit) There should be an edit mode. In edit mode the presentation can be changed.
- In edit mode:

[3] We might want that the aspect ratio, i.e. the vertical size divided by the horizontal size be a fixed number such an 9/16.

- (Ob-Add) It should be possible to add graphical objects to a presentation.
- (Ob-Delete) It should be possible to delete graphical objects from a presentation.
- There should be these kinds of graphical objects at least
  - (Ob-Text) Text areas.[4]
  - (Ob-Oval) Ovals.
  - (Ob-Line) Straight line segments.
  - (Ob-Rect) Rectangle.
  - (Ob-Image) Images. (*)
  - (Ob-Polygon) Other polygons. (*)
  - (Ob-Arc) Curved line segments (*).
  - (Ob-Path) Paths.(*)[5]
  - (Ob-Group) Groups. A group is an graphical object that contains other graphical objects. The position, size, and orientation of the group determines the position, size, and orientation of its members. (*)
- (Attributes) Graphical objects have attributes. The value of each attribute may depend on the state.
- Attributes should include, where appropriate.[6]
  - (Attr-Position) Position on the sheet.[7]
  - (Attr-Size) Size.
  - (Attr-Orient) Orientation. (*)[8]
  - (Attr-Outline-Colour) Outline colour (*)
  - (Attr-Thickness) Line Thickness (*)
  - (Attr-Fill-Colour) Fill colour.
  - (Attr-Text-Colour) Text colour.
  - (Attr-Text-Size) Text size.
  - (Attr-Text-Value) Text value.
  - (Attr-Visibility) Visibility.
- (State-Add) It should be possible to add a state at any point.

[4] A text area should be capable of showing several lines of text. The height and width of the text area should depend on text. For our minimal viable product, we can assume the text will not have any formatting. I.e., no italic, no bold, no changes of size or font, no bullet lists, numbered lists, tables, etc.

[5] A path could be a sequence of one or more straight (or possibly curved) line segments that are connected together.

[6] Certain attributes might not make sense for certain kinds of objects. For example position, height and width might not be the useful for lines, instead the positions of its endpoint and the line-width, might be more useful.

[7] If groups are implemented, the position, size and orientation might be with respect to the group.

[8] It would be nice if orientation were a continuous quantity, e.g. a floating point number in the interval $[0, 2\pi)$.

- (State-Delete) It should be possible to delete any state.
- (State-Reorder) It should be possible to reorder states. (*)
- (State-View) It should be possible to view the presentation at any state.
- (Attr-modify) It should be possible to reposition, resize, recolour, hide, reveal or generally change the values of the various attributes of graphical objects in the state currently being viewed.
- (Trans-Trigger) It should be possible to change the trigger condition for moving from one state to the next.
- Trigger conditions should include
  - (Trans-Trigger-Immediate) Immediately.
  - (Trans-Trigger-Timed) After a positive amount of time.
  - (Trans-Trigger-Key) Pressing a keyboard key such as the 'enter' key.
- (Trans-Duration) It should be possible to change the amount of time it takes to move from one state to the next. Zero seconds should be a possibility.
- It should be possible to change the way that interpolation during a transition happens. (*) Some possibilities are[9]
  - (Interp-Linear) Linear.
  - (Interp-EOEO) Ease-in-ease-out (i.e., sigmoid).
  - (Interp-Start) Step at start. (I.e. the attribute changes at the start of the transition.)
  - (Interp-End) Step at end. (I.e. the attribute changes at the end of the transition.)
  - (Interp-Middle) Step in the middle. (I.e. the attribute changes at the end of the transition.)
  - (Interp-Arc) Arc. (This one only makes sense for two dimensional attributes such as position.)
- (Interp-per-attribute) It should be possible to apply interpolation on a per object/attribute basis. (*)[10]
- (Edit-view-port) In edit mode it should be possible to zoom, pan, and orient the current view-port.
- (Camera) There should be a camera object whose attributes values in each state indicate the region of the sheet (and thus the objects) that will be seen during presentation of that state.

[9] The default should be linear or ease-in-ease-out for numeric attributes and middle for attributes where no sensible interpolation exists.

[10] This means that, for example, if two rectangles are moving at the same time, one could use linear interpolation for its position and the other could use step at start.

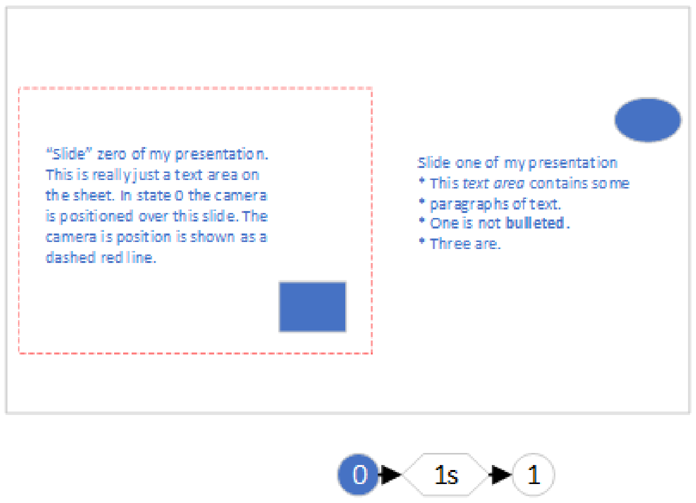
  - (Camera-Visible) There should be a visible representation of the camera's attributes. E.g., a rectangle that shows which part of the sheet the camera is showing.
  - (Camera-Attributes) The attributes of the camera include:
    - Position on the sheet.
    - Height and/or width. (The aspect ratio may be fixed.)
    - Orientation. (*)
  - (Pres-start) There should be a way to move from edit mode to presentation mode either starting at the beginning or starting at the current state.
  - (Undo-Redo) Edit mode should support undo and redo. (*)
  - (Cut-Copy-Paste) Edit mode should support cut, copy, and paste operations using the platform's clipboard. (*)

- There should be a presentation mode.
- In presentation mode:
  - (Pres-States) The user should be shown each state of the presentation in succession.
  - (Pres-trigger) The presentation begins moving to the next state when the appropriate trigger event happens.
  - (Pres-trans) Between showing one state and the next, a transition will be shown.
  - (Pres-interpolation-objects) While transitions between states are being shown, the user should see an interpolation of attribute values where possible.[11]
  - (Pres-interpolation-camera) Interpolation also applies to the camera attributes.
  - (Pres-skip) There should be a way to move quickly forward or backward through the presentation without time for transition, e.g., by using the arrow keys. (*)
  - (Pres-end) There should be a way to exit presentation mode and entering edit mode, e.g., by pressing the escape key.

2.6 Illustrations

Figure 1 shows the first state of a presentations with two states, as it might be seen in the editor. The camera is shown as a red-dashed rectangle. Figure 2 shows the second state; in this state, the camera has panned to the right, the rectangle has moved a bit to the right and

[11] When attributes have discrete values, 'interpolation' might not have an obvious meaning. For example, what string is half way between "love" and "bread"? What is half way between false and true? In these cases, you can come up with a creative approach to interpolation or the value should change in the middle of the transition.


“Slide” zero of my presentation. This is really just a text area on the sheet. In state 0 the camera is positioned over this slide. The camera is position is shown as a dashed red line.

Slide one of my presentation
* This *text area* contains some
* paragraphs of text.
* One is not **bulleted**.
* Three are.

Figure 1:

the oval has changed colour. Figure 3 shows the same thing, but now the user has zoomed in so that only the portion of the presentation that will be visible during presentation is shown in the editor. The three figures don’t show any modifications to the presentation: what is different between Figure 1 and Figure 2 is a change of temporal perspective; what is different between Figure 2 and Figure 3 is a change of spatial perspective.

Figure 4 shows the presentation at three points in time: while resting in the first state, part way through the transition to the second state, and while resting in the second state.

3 Nonfunctional requirements

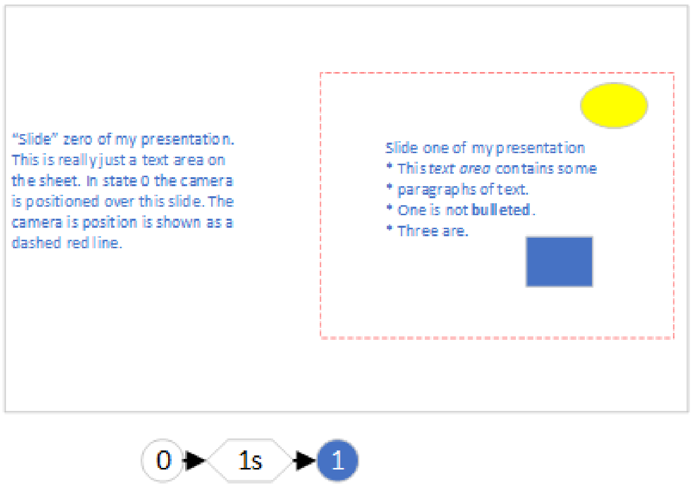
Language: You must program in one of these languages Java, Typescript, and Haxe.[12]

Platform: The system should be either a desktop application or a web-based application. (It could be both too, but please focus your time on one or the other.) Your program should run on one of the following platforms:

- The Java Runtime System, Standard Edition.
- Electron.
- Web-browsers. (In this case you will need a separate server component for storage of presentations.)

Testability: Your program should be accompanied by thorough unit tests for its parts. For this reason (and others), the design should strive to isolate dependences on the GUI

[12]I’ll consider others if you make a case for them. Likewise for platform. If Typescript is used, you should use it’s type-checking facilities to the fullest extent.


“Slide” zero of my presentation. This is really just a text area on the sheet. In state 0 the camera is positioned over this slide. The camera is position is shown as a dashed red line.

Slide one of my presentation
* This *text area* contains some
* paragraphs of text.
* One is not **bulleted**.
* Three are.

Figure 2:

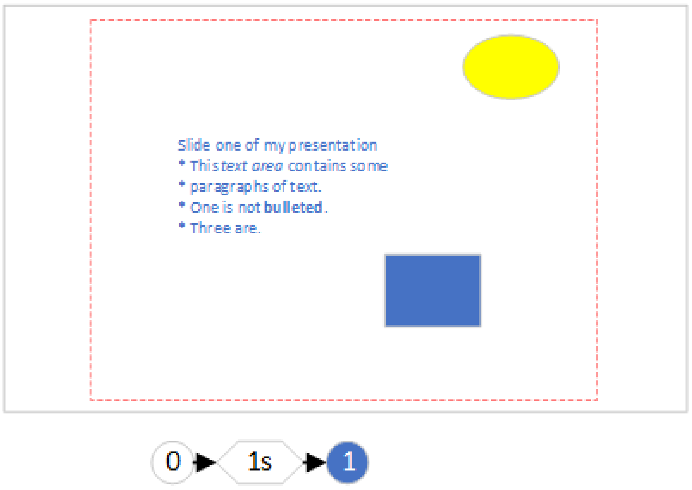
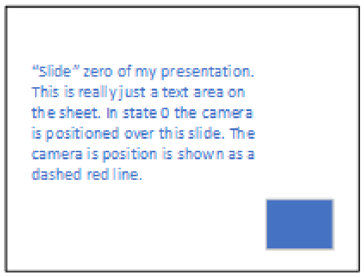
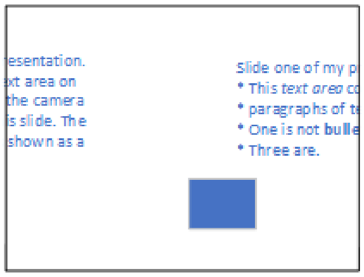

Slide one of my presentation
* This *text area* contains some
* paragraphs of text.
* One is not **bulleted**.
* Three are.

Figure 3:

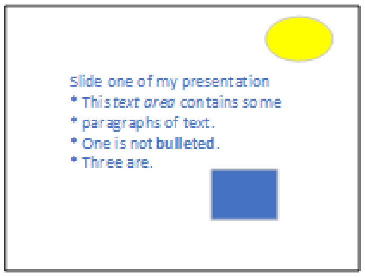


“Slide” zero of my presentation.
This is really just a text area on
the sheet. In state 0 the camera
is positioned over this slide. The
camera is position is shown as a
dashed red line.



esentation.
xt area on
the camera
is slide. The
shown as a

Slide one of my p
* This *text area* co
* paragraphs of te
* One is not **bulle**
* Three are.



Slide one of my presentation
* This *text area* contains some
* paragraphs of text.
* One is not **bulleted**.
* Three are.

Figure 4:

technology used. For example, it should possible to completely test the model and controller, without the presence of a GUI.

4 Some random thoughts and ideas

4.1 New states

When new states are created, it's a good idea for them to be created as copies of the preceeding state.

4.2 States transitions or both

In describing the system, I've tried to be noncommittal about how and where information about attributes values is stored. Consider this sequence of states and transitions that moves an object in a square

$$\overset{s_0}{\begin{pmatrix} r.x = 0 \\ r.y = 0 \end{pmatrix}} \overset{t_0}{\left(0 \xleftrightarrow{r.x} 100\right)} \overset{s_1}{\begin{pmatrix} r.x = 100 \\ r.y = 0 \end{pmatrix}} \overset{t_1}{\left(0 \xleftrightarrow{r.y} 100\right)} \overset{s_2}{\begin{pmatrix} r.x = 100 \\ r.y = 100 \end{pmatrix}}$$
$$\overset{t_2}{\left(100 \xleftrightarrow{r.x} 0\right)} \overset{s_3}{\begin{pmatrix} r.x = 0 \\ r.y = 100 \end{pmatrix}} \overset{t_3}{\left(100 \xleftrightarrow{r.y} 0\right)} \overset{s_4}{\begin{pmatrix} r.x = 0 \\ r.y = 100 \end{pmatrix}}$$

One way is to simply store all the information in states. The transitions can be largely inferred from the states

$$\overset{s_0}{\begin{pmatrix} r.x = 0 \\ r.y = 0 \end{pmatrix}} \overset{s_1}{\begin{pmatrix} r.x = 100 \\ r.y = 0 \end{pmatrix}} \overset{s_2}{\begin{pmatrix} r.x = 100 \\ r.y = 100 \end{pmatrix}} \overset{s_3}{\begin{pmatrix} r.x = 0 \\ r.y = 100 \end{pmatrix}} \overset{s_4}{\begin{pmatrix} r.x = 0 \\ r.y = 100 \end{pmatrix}}$$

Another way is to store the changes in the transitions. One problem that comes up is: where does the first state get its values? We can make a special transition at the start to solve this problem.

$$\overset{t_{-1}}{\begin{pmatrix} r.x := 0 \\ r.y := 0 \end{pmatrix}} \overset{t_0}{(r.x := 100)} \overset{t_1}{(r.y := 100)} \overset{t_2}{(r.x := 0)} \overset{t_3}{(r.y := 0)}$$

In this case the value in each state can be inferred from the preceeding transitions. There isn't a lot of difference between the approaches, perhaps.

- The second approach does make it easier to associate different interpolation strategies with each object/attribute/transition triple.

- Below I propose generalizing the idea of states having multiple outgoing and incoming transitions. In this case, the idea of associating changes with transitions becomes much more interesting because, depending on the sequence of transitions used to reach a state, the values of the attributes in the state might be different. A presentation is now a sort of finite state machine or flow chart. This allows presentations to have the computational power of a programming language.

4.3 Selection model

I haven't specified any notion of selecting objects. This is usually a good thing to be able to do in a GUI. I'll leave the details up to you.

4.4 Direct manipulation

For ease of editing, direct manipulation is often best. For example, one should be able to resize or reposition an object using only the mouse.

However for precision, it can be nice to be able to type values into a table. Also some operations have no obvious way of being expressed by direct manipulation, e.g., changing the style of a line from solid to dashed.

4.5 Editing the past, present, and the future

One interesting issue is what should happen to other states when an attribute value is edited? Suppose we have a sequence of 5 states and 4 transitions

$$s_0\ t_0\ s_1\ t_1\ s_2\ t_2\ s_3\ t_3\ s_4$$

and suppose the horizontal position of r is 0 in states s_0, s_1, s_2, and 50 in state s_3 and s_4.

$$\begin{array}{ccccccccc} s_0 & t_0 & s_1 & t_1 & s_2 & t_2 & s_3 & t_3 & s_4 \\ (r.x = 0) & & (r.x = 0) & & (r.x = 0) & \left(0 \xleftrightarrow{r.x} 50\right) & (r.x = 50) & & (r.x = 50) \end{array}$$

In state s_1 the user drags r 100 pixels to the right. What states should be affected and how? Here are some possibilities.

1. The position of the rectangle is changed to 100 in all states.

$$\begin{array}{ccccccccc} s_0 & t_0 & s_1 & t_1 & s_2 & t_2 & s_3 & t_3 & s_4 \\ (r.x = 100) & & (r.x = 100) & & (r.x = 100) & & (r.x = 100) & & (r.x = 100) \end{array}$$

2. Only state s_1 is affected.

$$\begin{array}{ccccccccc} s_0 & t_0 & s_1 & t_1 & s_2 & t_2 & s_3 & t_3 & s_4 \\ (r.x = 0) & \left(0 \xleftrightarrow{r.x} 100\right) & (r.x = 100) & \left(100 \xleftrightarrow{r.x} 0\right) & (r.x = 0) & \left(0 \xleftrightarrow{r.x} 50\right) & (r.x = 50) & & (r.x = 50) \end{array}$$

3. The present and all future states are affected. For s_1 and all future states the position is set to 100. s_0 is left alone.

$$\begin{array}{ccccccccc} s_0 & t_0 & s_1 & t_1 & s_2 & t_2 & s_3 & t_3 & s_4 \\ (r.x = 0) & \left(0 \xleftrightarrow{r.x} 100\right) & (r.x = 100) & & (r.x = 100) & & (r.x = 100) & & (r.x = 100) \end{array}$$

4. The present and all future states are affected, but the changes are cumulative. For s_1 and s_2 the position will be set to 100 and for s_3 and s_4 it is set to 150. s_0 is left alone.

$$\begin{array}{ccccccccc} s_0 & t_0 & s_1 & t_1 & s_2 & t_2 & s_3 & t_3 & s_4 \\ (r.x = 0) & \left(0 \xleftrightarrow{r.x} 100\right) & (r.x = 100) & & (r.x = 100) & \left(100 \xleftrightarrow{r.x} 150\right) & (r.x = 150) & & (r.x = 150) \end{array}$$

5. The present and all future states are affected up to the next transition that affects the same attribute of the same object. For s_1 and s_2 the position will be set to 100 and for s_3 and s_4 it is set to 50. s_0 is left alone.

$$\begin{array}{ccccccccc} s_0 & t_0 & s_1 & t_1 & s_2 & t_2 & s_3 & t_3 & s_4 \\ (r.x = 0) & \left(0 \xleftrightarrow{r.x} 100\right) & (r.x = 100) & & (r.x = 100) & \left(100 \xleftrightarrow{r.x} 50\right) & (r.x = 50) & & (r.x = 50) \end{array}$$

Different choices might be appropriate for different situations. Note that method 1 can be simulated by 3 by making the change on the first state. Method 2 can be simulated by 3, 4, or 5 by applying a change in one state and then its opposite in the next state. Method 4 only makes sense when values are numeric; so another strategy could be to use 4 when it makes sense and otherwise to use 5.

A similar issue comes up when objects are created. It is probably best if the new object is invisible in earlier states and visible in the current and all later state.

4.6 Future directions

- Multiple sheets. If there is only one sheet, then different "slides" need to be in different parts of the sheet.

- It would be nice for objects to have "connection points" that can connect objects. For example the end of a line or path might be connected to a point on an oval or not, depending on the states. Then when the oval moves, so does the end point. This is very handy for drawing graphs.

- Nonlinear presentations. It would be interesting to generalize the idea of a sequence of states and transitions to allow transitions to connect any two states. This way we essentially have a finite state automaton. This means that some states would have multiple outgoing transitions that would have different triggers. (For example, a click on one rectangle triggers one transition, while a click on another triggers another transition.) This could mean that there might be multiple paths from the start state to each state that imply different values for the attributes, which leads to the interesting question of how a state should be shown during edit time and how changes to an attribute value during editing would affect the transitions. This generalization leads to an interesting form of "programming by example". It could be really interesting.

- Fancy text. Of course presentation software should have good support for rich text: Within a text area, we would like to be able to use multiple fonts and text styles, paragraph styles, tables, mathematical typesetting, etc. etc. The sky is the limit. This is a very big topic. Note that Java does have a rich text format (rtf) editor and a primitive HTML editor in its library that could of use. I'd be content to leave fancy text as an area for future expansion.

- It might be interesting for the user, during editing, to be able to preview a transition.