Technical Note TN2073
Deconstructing A Keynote Document: Part One - Slides

CONTENTS

This is the first of four Technical Notes describing the Keynote XML document format (referred to here and elsewhere as the "APXL schema" or simply "APXL"). Our approach is to use a sample Keynote presentation and then examine the XML that is generated. In this first document, we'll give a brief overview of the whole file format, and then discuss in detail the portion of the XML that describes the list of slides, focusing on Keynote's basic text and graphics functionality. Later documents will focus on plugins (including movies, charts and tables), themes, and advanced functionality.

[May 20, 2003]

Keynote is Apple's new presentation software that makes it easy to create compelling presentations. The primary format for Keynote documents is XML. As part of its commitment to openness, Apple has released the Keynote XML specification, allowing developers to create entire presentations -- with text, images, transitions and more -- programmatically, outside of Keynote. This means you can build dynamic, always-up-to-date presentations by, for example, pulling the latest sales figures from a database or grabbing data from a variety of Internet sources, and creating new presentations from scratch or populating existing ones.

Our approach is to use a sample Keynote presentation and then examine the XML that is generated. Each section of this document builds on the previous section, so we recommend you read through the entire document at least once. We also recommend you go through the included sample presentation at least once, and keep it handy as you read through this document.

IMPORTANT:
This series of Technical Notes is not a formal specification of the file format. Please see Technical Note 2067: About the Keynote XML File Format (APXL Schema) [1] for the actual APXL specification. This series will also not feature any specific programming code, language or tool. Instead, descriptions will be given in plain English. Although this means you can't simply drop the examples into your favorite development environment and run them, it also means you'll be able to take what you learn here and apply it to writing code in whichever language you want, using the tools of your choice, including a plain text editor to handcraft the XML. If you'd like to see code for generating a APXL document, there is a command-line tool (PictureShow) [2] that shows how to create a presentation using Java. You may use that code as a starting point for your own efforts.

Note:
If you're interested in using or creating open-source software that works with the APXL schema, please join the Keynote tools mailing list and project at Open Darwin [3].

The Document Bundle

Keynote documents are actually bundles [4] -- specially constructed folders that the Finder knows to display as single files. This offers two advantages: users only have to worry about a "single file"; while developers can easily modify the individual portions of the bundle as necessary.

You can examine the contents of a bundle by Control-clicking (or right-clicking with a two-button mouse) on a Keynote document and selecting "Show Package Contents" from the resulting contextual menu.

The bundle contains several files and subdirectories:

- presentation.apxl: This required file is the XML document and describes how a Keynote document looks and acts.
- media files (images, sounds, movies, etc.) referenced by the presentation;
- .typeAttributes.dict, which tells the OS this folder should be treated as a bundle;
- tiny: a folder containing small versions of all the images in the document
- thumbs: a folder containing "thumbnails" of all the slides in the document.

The two folders ("tiny" and "thumbs") provide performance improvements for the application. However, you can ignore them completely -- if you programmatically create a Keynote document without these folders, Keynote will open the document correctly; when you save the document in the application, Keynote will fill in the correct values for you.

Back to top

About the presentation.apxl file

The center of a Keynote document bundle is the "presentation.apxl" file. Using the sample presentation, we'll walk through this file section by section. The top-level element in a presentation.apxl file is `presentation` . It has one attribute, `version` . Keynote 1.0 always writes a value of "36" to this field. You should also write a value of "36". If you do something else, Keynote may refuse to read your document, or do strange and unexpected things when it does read it.

The `presentation` element has four children:

- metadata
- theme
- slide-list and
- ui-state.

You can skip the `metadata` element entirely and the document will load just fine. Or you can add data to it to help identify the source of your document to track your internal workflow. (Keynote will provide its own metadata when the application saves the document for the first time.)

The `theme` describes the default appearance of the document as a whole and each master slide. It will be covered in detail in the third Technical Note in this series. To start, you can just use the theme specification from one of the default themes that ships with Keynote (or your own custom theme). In fact, Keynote themes are a special version of Keynote documents, and you can create a Keynote document by simply loading a theme and then populating the slide-list element.

The `ui-state` element describes the user interface state of a document; it is required, but you can leave it empty if you wish. The `ui-state` allows you to do things like set which slide is the first slide the user sees when they open the document, the size of the document window, or the zoom level of the document. It is easy to reverse-engineer; we will discuss it further the fourth part of this series.

The heart of a presentation is the `slide-list` element. It consists of a list of slides. Absolutely nothing else can appear in a `slide-list` . The order of slides in the list is the same as their order of appearance in the presentation. A Keynote document has to contain at least one slide.

Back to top

The Basics of a Slide

If you look at the sample document, the first slide is as basic as you can get: it's completely blank. However, looking at the XML reveals that there's a bit of complexity hidden here:

Listing 1. XML from Slide 1

```
<slide master-slide-id="master-slide-2">
  <drawables>
    <body visibility="tracks-master" vertical-alignment="tracks-master"/>
    <title visibility="tracks-master"
     vertical-alignment="tracks-master"/>
  </drawables>
  <transition-style type="inherited"/>
  <thumbnails>
    <thumbnail file="thumbs/st0.tiff" byte-size="360" size="60 45"/>
  </thumbnails>
  <bullets>
    <bullet marker-type="inherited" level="0">
      <content tab-stops="L 96" font-size="84" font-name="GillSans"
       paragraph-alignment="center"/>
    </bullet>
  </bullets>
</slide>
```

The `slide` element in this case has a single attribute, the `master-slide-id`. This determines which master slide the slide uses for default values. If you look at the master slides in the theme element of the presentation, you'll see that the `master-slide` element with id "master-slide-2" corresponds to the master slide named "Title & Bullets" in the Keynote UI for this theme.

In a presentation.apxl file, you can change which master slide a slide uses by changing the `master-slide-id` attribute. It must, of course, reference a `master-slide` in the document's theme. Every slide element must have a `master-slide-id` attribute with a valid value.

Now let's consider the children of the slide element. The first of these is `drawables` . Even though this slide is blank, there are two objects in its `drawables` list, a `title` and a `body` . These correspond to the title and body placeholders on the slide canvas in Keynote. Both have two attributes, `visibility` and `vertical-alignment`, which are both set to "tracks-master" in this case. This means that if these things are visible on the master slide, they'll be visible on the slide, and they inherit their vertical alignment style from the master slide as well. We'll show examples of different visibility and vertical alignment examples below.

The next child is `transition-style`, and its attribute `type` has the value "inherited". This means that this slide inherits whatever transition its master slide has. When we look at the next slide, we'll see an example of overriding this.

Next is `thumbnails` . Remember, the "thumbs" directory in the document bundle exists to provide a performance boost to Keynote. The slide thumbnails correspond to the images you see in Keynote's Slide Navigator. You don't have to have them in the document; if you don't, Keynote will generate them when the document is opened. You can leave the `thumbnails` element empty, or you can add as many thumbnails as you want. The `file` attribute tells Keynote where to locate the image for the thumbnail -- in this case, "thumbs/st1.tiff". Thumbnails don't have to be tiffs, and they don't have to be in the thumbs subdirectory, but we recommend you put them there anyway -- it'll make your documents more like everyone else's, and it makes it easier for others to read the document bundle.

The `byte-size` attribute is also there for performance reasons. If you want to create a thumbnail, and you don't know what the file size it's going to be, set this to "-1", otherwise set this value to the file size (in bytes).

The `size` attribute describes the dimension of the thumbnail in pixels. The first value is width, the second is height. To get the performance enhancement in the slide navigator, create a thumbnail of size 60x45.

Remember, you don't have to create thumbnails. You can leave this element empty and Keynote will work just fine (and will generate the correct information when you open the document in Keynote).

The last child of the slide element is `bullets` . Every slide has a `bullet` list, and every `bullet` list contains at least one bullet, even if that bullet has no contents, as is the case here. We'll discuss bullets in more detail below.

Back to top

Adding a Shape and a Transition

The second slide in the sample presentation contains an arrow, and has a mosaic transition to the third slide. Examining the XML, we can see how these are encoded:

Listing 2. XML from Slide 2

```
<slide id="slide-1" master-slide-id="master-slide-4">

  <drawables>

    <shape path="M 0 33.3 L 0 66.7 L 60 66.7 L 60 100 L 100 50 L 60 0 L 60

     33.3 Z M 0 33.3" transformation="1.39564 0 0 1.07636 442 330.364">

    <title visibility="tracks-master"

     vertical-alignment="tracks-master"/>

    <body visibility="tracks-master" vertical-alignment="tracks-master"/>

  </drawables>

  <transition-style type="apple:3D-mosaic-small" duration="1.5">

    <property name="direction" value="13"/>
```

```
        </transition-style>

        <thumbnails>

          <thumbnail file="thumbs/st1.tiff" byte-size="478" size="60 45"/>

        </thumbnails>

        <bullets>

          <bullet marker-type="inherited" level="0">

            <content tab-stops="L 96" font-size="84" font-name="GillSans"

             paragraph-alignment="center"/>

          </bullet>

        </bullets>

      </slide>
```

First, note that this slide uses a different master slide -- "master-slide-4", which in this Keynote document is presented to the user with the name "Blank".

Let's skip the `drawables` tag for the moment and instead examine the `transition-style` tag next. This is where the transition for a slide is encoded. On the first slide, it took the master slide's transition. Here we have overridden the master slide.

The `transition-style` tag has two attributes, `type` and `duration` . The `type` attribute is a string naming the transition, in this case "apple:3D-mosaic-small". The duration attribute tells the application how many seconds the transition should last -- in this case, 1.5 seconds. This must be a non-negative number. If it's too small, the transition will be too fast for the viewer to perceive. If it's too large, the transition will play very slowly.

In this case the `transition-style` tag has one child, a `property` tag. Each transition has zero or more properties, all of which are expressed through a `property` tag. Each `property` tag has two attributes, the property's `name` -- in this case, "direction" -- and the property's `value` -- in this case, "13", which in Keynote translates to "down." Different transitions have different sets of style attributes and potential values:

Table 1. Acceptable values for the "transition-style" tag.

Transition type	Description	Attributes and Values
apple:3D-cube	Draws the current and next slide on the faces of a 3D cube, then rotates the cube	"Direction" -- the direction of the rotation 11 - right to left 12 - left to right 13 - bottom to top 14 - top to bottom
apple:revolve	Draws the current slide on one side of a "sheet", and the next slide on the other side, and then flips the "sheet" around the center	"Direction" -- the direction of the rotation 10 - left to right 11 - right to left 12 - top to bottom 13 - bottom to top
apple:3D-mosaic-large	Draws the current slide on one side of a "sheet" and the next slide on the opposite side of the sheet, then divides the sheet into "tiles" and flips the tiles individually	"Direction" -- describes both the direction that the tiles flip, and the order in which they are flipped 11 - right to left 12 - left to right 13 - bottom to top

		14 - top to bottom
apple:3D-mosaic-small	Draws the current slide on one side of a "sheet" and the next slide on the opposite side of the sheet, then divides the sheet into "tiles" and flips the tiles individually; the tiles are smaller than for apple:3D-mosaic-large	"Direction" -- describes both the direction that the tiles flip, and the order in which they are flipped 11 - right to left 12 - left to right 13 - bottom to top 14 - top to bottom
apple:dissolve	A dissolve effect between the current and next slide	no attributes
apple:bounce	The next slide drops in front of the current slide, bouncing a few times	no attributes
apple:fade	Fades through black to the next slide	no attributes
apple:motion dissolve	The current slide "moves out" while dissolving out, and the next slide "moves in" while dissolving in	no attributes
apple:slide	The next slide moves in to cover the current slide	"Direction" -- describes both the direction that the next slide moves 10 - left to right 11 - right to left 12 - top to bottom 13 - bottom to top 14 - top left to bottom right 15 - top right to bottom left 16 - bottom left to top right 17 - bottom right to top left
apple:pivot	The next slide pivots around one of the four corners to cover the current slide	"Direction" -- describes the direction from which the pivot comes; you can think of it as describing the corner around which the pivot occurs 10 - top left 13 - top right 14 - bottom right 17 - bottom left
apple:push	The next slide slides in while the current slide slides out, as if the next slide was pushing the current slide out of the way	"Direction" -- describes the direction in which both slides move 10 - left to right 11 - right to left 12 - top to bottom 13 - bottom to top
apple:reveal	Moves the current slide away, revealing the next slide underneath	"Direction" -- describes both the direction that the current slide moves 10 - left to right 11 - right to left 12 - top to bottom 13 - bottom to top 14 - top left to bottom right 15 - top right to bottom left 16 - bottom left to top right 17 - bottom right to top left
apple:scale	Scales the current or the next slide and provides a dissolve-style effect between them	"Direction" -- describes which slide scales and how 30 - up - the next slide

		starts small and scales up until it is the right size 31 - down - the current slide scales down 32 - out - the current slide scales up while it dissolves out 33 - in - the next slide starts larger and scales down until it is the right size
apple:twirl	The current slide twirls away while scaling down, and then the next slide twirls in while scaling up	no attributes
apple:wipe	A cinematic wipe between the current and next slide	"Direction" -- describes the direction that the wipe effect moves 20 - left to right 21 - right to left 22 - top to bottom 23 - bottom to top

Let's return to the `drawables` tag. Note that even on a blank slide, we have `title` and `body` drawables. Note also that (encoded in the `bullets` element) we have a bullet point. Bullets always go in the `body` placeholder, and our `body` placeholder has a visibility of `tracks-master` (as does the `title` placeholder.) On the master slide, both placeholders are set to "invisible".

> **Note:**
> Why do this? It is more of a UI issue than a file format issue: this way, the user can experiment with different master slides, and not lose the text content if she tries a blank slide, and then a slide with a visible placeholder.

The first drawable is a shape . The shape tag has two attributes, `path` and `transformation` . The `path` attribute describes a path to draw the shape, and then the `transformation` describes scaling, rotation, translation and skew of that object.

Keynote always generates paths based on a 100 pixel by 100 pixel square, and they are always aligned to the origin. Shapes must always be aligned to the origin, or Keynote will not handle them correctly. However, you needn't follow Keynote's lead as far as the 100x100 pixel square; you can create shapes using any size path, as long as the top left corner of the square enclosing the shape is at the origin.

The path description in the Keynote XML is similar to a path description in the Scalable Vector Graphics (SVG) [5] standard. A Keynote path consists of any number of steps, and each step can be an (M)ove-step, a (L)ine-step, a (C)urve-to-step, or a clo(Z)e-step. To understand these, imagine that you are drawing the shape with a pen on a piece of paper:

Table 2. Path description commands

Step Type	Format	Description
M-step	M x y	Move to - pick the pen up off the piece of paper, and move it to the (x, y).
L-step	L x y	Line to -- put the pen down (if it's not already down) and draw a line to the (x, y).
C-step	C x1 y1 x2 y2 x y	Curve to -- put the pen down (if it's not already down) and draw a curve from the current point to (x, y), using (x1, y1) and (x2, y2) as "control points", which determine the shape of the curve.
Z-step	Z	cloZe path -- put the pen down (if it's not already down) and draw a straight line from the current point to the first point in the path.

You should start each path with a `move-to`, and then use `line-to` and `curve-to` to draw the path you wish. If you are drawing a polygon -- or any time you wish your last segment to be a straight line -- you must close with a `Z` .

You may notice that the set of paths that can be expressed is far greater than the set of drawing tools that Keynote provides. You can, in fact, create shapes in the APXL that you cannot create through the user interface, if you find it useful to do so. Just make sure they are aligned to the origin, and that they're closed.

Drawing the path is only part of the story. We also need to position the shape. This is done with the `transformation` attribute. The `transformation` attribute consists of six numbers that specify the six variables of an affine transform matrix:

Table 3. Affine transformation matrix

m11	m12	0
m21	m22	1
tx	ty	1

These are specified in the attributes string as "m11 m12 m21 m22 tx ty". Since the right-most column never changes, we don't bother to specify it.

Before a shape is drawn, it is transformed by this matrix. The result is that any point (x,y) on the untransformed path becomes (x', y') on the transformed path, where `x' = (m11)x + (m21)y + txy' = (m12)x + (m22)y + ty`

Note:
For more information on the affine transformation matrix, see Basic Drawing: Coordinate Transforms [6]

Recall that the path must always be aligned to the origin, yet shapes can appear anywhere on the screen. This is because we use the affine transform to translate the shape. Also recall that Keynote always draws its paths in a 100x100 pixel box, but paths can be any size. This is because the shapes are scaled by the affine transform matrix. The path for a Keynote shape is always unrotated, and rotation is applied through the transformation matrix.

Note:
It is also possible to apply shearing to a shape using the transformation matrix, although Keynote's user interface does not support shearing. We recommend caution when you experiment with shearing. While it may display correctly, the user interface it presents may be incorrect or confusing. If you expect anyone to edit your document by hand, it's probably best to avoid shearing altogether.

Back to top

Multiple Drawables and Overriding Shape Appearance

`Path` and `transform` are only two parts of the shape's appearance. When you see it on the screen, it also has a particular fill, a line drawn around it, and it may be solid or semi-transparent. Note that in our example above, none of these parameters were specified. Those parameters were specified by the master slide or the theme, and since we didn't override them, Keynote just used the default parameters. This section describes how to override those parameters. We'll also look briefly at how Keynote handles the layering of objects

Slide 3 of our sample document uses a blank master slide and has two shapes on it, one right on top of the other. It's just like Slide 2, except that its `drawables` element has different children, so let's just consider that element:

Listing 3. XML from Slide 3

```
<drawables>
  <shape path="M 0 100 L 100 100 L 50 0 Z M 0 100" transformation="1 0 0 1
   462 334">
    <styles>
      <shadow-style opacity="0.75" angle="130" offset="24" radius="10"
       color="0 0.252033 0"/>
    </styles>
  </shape>
  <shape stroke-color="1 0 0.498039" path="M 85.3553 14.6447 C 104.882
   34.1709 104.882 65.8291 85.3553 85.3553 C 65.8291 104.882 34.1709
   104.882 14.6447 85.3553 C -4.88155 65.8291 -4.88155 34.1709 14.6447
   14.6447 C 34.1709 -4.88155 65.8291 -4.88155 85.3553 14.6447"
   opacity="0.473684" stroke-width="5" transformation="1 0 0 1 461 345">
    <styles>
      <fill-style fill-color="0.25098 1 0.25098" fill-type="color"/>
      <dash-style pattern="6 6"/>
    </styles>
  </shape>
  <title visibility="tracks-master" vertical-alignment="tracks-master"/>
  <body visibility="tracks-master" vertical-alignment="tracks-master"/>
</drawables>
```

Note that, again, the `title` and `body` drawables appear, and again they track their master slide.

The first shape is the triangle we see on the slide, and the second shape is the circle that is on top of the triangle. Keynote always lists drawables from bottom to top: the lowest drawable is first, the highest is last. Thus, the circle is on top of the triangle visually, and is listed after the triangle in the `drawables` list.

Let's focus on the circle first. Note that it provides an example of the `C-step` path element discussed previously. The shape tag for the circle also has three new attributes, `stroke-color`, `stroke-width` and `opacity`, and one new child element, `styles` .

The `stroke-color` attribute describes the color of the line around the shape. It's an RGB value, with 0 being the complete absence of the color, and 1 being as much of the color present as possible. Here we see a lot of red, no green, and about half the possible blue. This color description format appears in other places; we'll see it again when we look at the fill color, shortly. Values for each color in `stroke-color` must be between 0 and 1, inclusively.

The attribute `stroke-width` describes how many pixels wide the stroke is. The value must be a positive integer. You can control whether any stroke is drawn around the figure using the dash style, described below.

IMPORTANT:
You'll note that you can set `stroke-width` to 0. Doing so will draw the object without a stroke, as expected. However, you can't set the stroke width to 0 in Keynote, leading to possible user confusion when she makes changes to the related dash styles and sees no change. The preferred way of drawing an object without a stroke is to use the `dash-style` element (described below) with a pattern of "none".

The `attribute` opacity specifies how opaque the object is, with 1 being completely opaque and 0 being completely transparent. You can see that this shape is about 47% opaque according to the file format, and if you look at the slide in the application, you'll see that you can in fact see through it to the triangle underneath.

Next is the `styles` element. Style information for most Keynote objects is stored in one of two places: the attributes of the object, or a `styles` child element. In this case, our `styles` element specifies a `fill-style` and a `dash-style` .

The `fill-style` element has two attributes. The `fill-type` attribute states whether it is a color fill, a gradient fill, or an image fill. Here we use a color fill, and in the `fill-color` attribute, we state what color it should be, using the same RGB format we used for `stroke-color` above. We'll see how to set up gradient and image fills below.

Let's examine the `dash-style` . It has one attribute, `pattern` , which in this case has a value of "6 6".

If you set the `dash-style` pattern attribute to have a value of "none", then there is no line drawn around the figure. If you set it to be "solid", then there is a solid line drawn around the figure. Here we have a more complex style, consisting of several pixels drawn solid, followed by several pixels blank. The numerical pattern "6 6" specifies the length of the solid and blank portions.

Note, however, that the solid and blank portions aren't 6 pixels long. They're 30 pixels long. Recall that the stroke width is 5 pixels. Keynote multiplies the values of the dash style pattern and the stroke width to determine how many pixels to draw or skip. Then, if the user changes the stroke width, the program automatically modifies the appearance of the dash style to match the new width.

You can specify up to 3 pairs of numbers in the dash style pattern attribute, which allows you to create fairly complicated dash styles. The numbers need to be positive, but they don't need to be integers -- so if you have a `stroke-width` of 10 pixels but you really need just one pixel on, you can use a value of 0.1.

Let's turn our attention back to the triangle. It has a style that the circle lacks: a shadow. This is encoded with the `shadow-style` child of the styles tag. The `shadow-style` tag has five attributes.

The `opacity` and `color` attributes should look familiar: they describe the opacity and color of the shadow, in the same way that the `opacity` attribute on the circle describes its opacity, and the `stroke-color` attribute on the circle describes the color of the circle's stroke.

The `angle`, `offset` and `radius` attributes are specific to the shadow style. The `angle` attribute describes the angle of the shadow in degrees, from 0 but less than 360. The `offset` attribute says how far away from the original object the shadow is, in pixels. The `radius` attribute describes the radius of the blurring effect, in pixels; if the user sets the blur to "1", then the radius will have a value of "1" and the shadow will appear very sharp; the higher the value, the less distinct the shadow.

Back to top

Images and Free Text

We've seen how to draw shapes and how to make transitions between slides. Let's add some more content, in the form of text and images.

Consider the drawables array for slide 4:

Listing 4. XML from Slide 4

```
 <drawables>

   <image display-name="Scale_Even.tif" byte-size="479568" natural-size="623
```

```
     768" image-data="Scale_Even.tif" transformation="0.78125 0 0 0.78125 268
     84" lock-aspect-ratio="true"/>
    <textbox id="textbox-1" grow-horizontally="true" transformation="1 0 0 1
     47 38.5" size="308 199.239">
      <content tab-stops="L 84" font-size="42" font-name="GillSans"
       paragraph-line-spacing="26.1194" paragraph-alignment="center">
        <span font-name="GillSans-Italic">This</span>
        <![CDATA[ is a]]>
        <span font-kerning="43.2836" font-name="GillSans-Bold">scal</span>
        <span font-name="GillSans-Bold">e</span>
        <![CDATA[.<&>]]>
      </content>
      <styles>
        <shadow-style opacity="0" radius="0"/>
      </styles>
    </textbox>
    <title visibility="tracks-master" vertical-alignment="tracks-master"/>
    <body visibility="tracks-master" vertical-alignment="tracks-master"/>
  </drawables>
```

The first child of the `drawables` array is an image of a balance scale. The attributes specify everything Keynote needs to display the image. Let's take them one at a time.

The first attribute, `display-name`, specifies the name of the object as it should appear to the user in Keynote's Metrics inspector. You can specify any value you want for this -- it's there for the user's convenience.

The next attribute is `byte-size` and it provides a performance enhancement to Keynote. If you're creating a document from scratch, and you don't know the size of the image in bytes, use a value of "-1" and Keynote will fill in the correct value when it opens the document.

The `natural-size` attribute describes the original ("natural") width and height of the image before scaling. These provide Keynote with "hints" that are useful if the document becomes corrupt and the image file cannot be found. If you don't include this attribute, or give it incorrect information, and then somehow the image disappears (for instance, if a download fails and the document is not completely transferred), then Keynote may not display the slide properly.

The `image-data` attribute provides the name of the image file in the document bundle. When you create a presentation by hand (or from another program), you must make sure to include the image in the document bundle.

The `transformation` attribute is just like the `transformation` tag on `shape` objects, and we discussed it in some detail above .

The `lock-aspect-ratio` attribute is a boolean tag. It tells Keynote how to treat the image when the user resizes it in the Metrics inspector. If this value is "true", then width and height will be locked to the same aspect ratio; if it's "false", then the user can scale width and height independently.

Though they aren't included in this example, the `image` tag can also take an `opacity` attribute and a `stroke-width` and `stroke-color` attribute. These act just like the attributes of the same name on the [[`shape` tag]]]. Also, the `image` tag can have a `styles` child tag, in which you can specify shadow styles and dash styles, just like with [[[shapes]]].

The second drawable on this slide is a textbox. The contents of the `textbox` tag describe the text within it, while the attributes describe the positioning and behavior of the textbox. We'll examine the attributes first.

The first attribute is an `id` attribute, with a value of "textbox-1". This kind of attribute is used when we need to refer back to a drawable later on in the file, either to encode user interface state or to provide information about builds. We'll come back to this attribute when we discuss builds in the next section .

The second attribute, `grow-horizontally`, is a boolean attribute that describes how the textbox should behave when the user edits it. If it's set to "true" (as it is here), then the textbox can grow in width as the user types; otherwise it cannot.

The `transformation` attribute describes how to scale, rotate, and translate the textbox, just as we've seen for previous objects. The `size` attribute describes the width and height of the textbox, in pixels. It's important to set this accurately, since the text you describe must fit in the size given to be visible.

The `content` child describes the text content of the textbox. The attributes define the default styling for all of the text -- in this case, it describes one tab stop, and default font of GillSans in 42-point, a center paragraph alignment, and a line-spacing of 26.11 points.

Any text that overrides that styling should be in a `span` tag as a child of the `content` tag. For example, the first word of the text box is "This" in italic type, and so we create a new `span` setting the `font-name` to "GillSans-Italic."

The next two words, "is a", are enclosed in a `CDATA` section. This is not strictly necessary, but it's a good idea -- it lets Keynote know that everything before the close of the `CDATA` section should be treated as text, not as an XML tag. Thus, you could have text such as "</drawables>" without destroying the integrity of the `drawables` array. You can place any text within a `CDATA` section.

The next two `span` s show how the word "scale" is made bold, with large kerning between the letters. Note that you cannot include a `span` within a `span` -- in order to make the "e" bold but not have the same kerning, we have to put it in a separate `span` . Using combinations of attributes, you can construct very complex text styles.

Free-standing text isn't the only way to put words on a slide -- remember, we still have those placeholder objects to play with. As we'll see below, once you understand how to specify the contents of a textbox, specifying the contents of a bullet point is pretty easy -- they use the same structure and format.

Back to top

Builds

If you play slide 4, you'll see that the text doesn't simply sit on there: it drops in, and then sort of falls off the left side. This is because the text has builds applied to it.

Recall that the textbox had a new attribute, `id`, with a value of "textbox-1". Take a look at the `events` child of the fourth slide:

Listing 5. XML from Slide 4

```
<events>

  <build target-id="textbox-1" type="apple:dropbuild" duration="2.5">

    <property name="animationType" value="1"/>

  </build>

  <build target-id="textbox-1" type="apple:pivot-build" duration="0.5">

    <property name="direction" value="14"/>

    <property name="animationType" value="2"/>
```

```
          </build>

        </events>
```

The `events` tag specifies all of the build events that occur on a slide. The order they appear in the XML is the order they appear during presentation. Each build is specified by a `build` element.

The `target-id` tag of a build event determines which object the event applies to -- in this case, the object with the `id` "textbox-1", or the textbox that appears on this slide. In order to build an object, then, you must have an `id` attribute on it -- and it must be on the same slide as the build event.

The `type` attribute describes the kind of animation that should take place. It's a string specifying which animation to use -- in this case, "apple:dropbuild" for the first build, and "apple:pivot-build" for the second. The `duration` attribute is the amount of time, in seconds, the build takes. It should be a non-negative number, and the same caveats apply to it that applied to transitions -- too fast, and it'll happen before the user notices; too slow, and it'll look odd.

A `build` element can have several properties, each of which has a `name` and a `value` . The drop build and the pivot build both have a property with the `name` "animationType". Every build must have such a property. If it has a value of "1", then the build brings the object onto the slide; a value of "2" means the build takes the object off of the slide. You can only build a single object onto the slide once, and similarly you can only take it off of the slide once; Keynote may behave a bit strangely if you try to violate this constraint.

The second build (the pivot) has another property, the `direction`, which tells the application which direction the build should move in.

If you play the fourth and fifth slides of the presentation, you'll notice a subtle difference between them: for slide 4, the presenter has to click the mouse or press a key for the text to build, but for slide 5 it happens automatically. The only difference between these slides is the `slide` tag: on slide 5, it has an additional attribute, `autobuild` , which has a value of "true". This tells Keynote to play the first build without action on the part of the user. If you leave this attribute absent, or you set its value to "false", then Keynote requires an action to play the first build.

Keynote provides a number of builds, some of which have attributes that describe exactly how they'll look:

Table 4. Acceptable values of the "Build" tag's "type"

Build "Type"	Description	Attributes and Values
apple:appear	the specified object simply appears or disappears	animationType -- whether the object appears or disappears 1 - appear 2 - disappear
apple:dropbuild	The specified object drops into the slide and bounces a few times	animationType -- whether the object moves in or out 1 - the object dissolves in 2 - the object dissolves out direction -- specifies the direction of the revolution 17 - left to right 23 - right to left 85 - top to bottom 93 - bottom to top
apple:move in	The specified object moves in or out	animationType -- whether the object moves in or out 1 - move in 2 - move out direction -- which way the object moves 0 - left to right in, right to left out 1 - right to left in, left to

		right out 2 - top to bottom in, bottom to top out 3 - bottom to top in, top to bottom out 4 - upper left to lower right in, lower right to upper left out 5 - upper right to lower left in, lower left to upper right out 6 - lower left to upper right in, upper right to lower left out 7 - lower right to upper left in, upper left to lower right out
apple:pivot	The specified object pivots in or out, as if attached by only one of its corners	animationType -- whether the object pivots in or out 1 - in 2 - out direction -- specifies the corner around which the pivot occurs 10 - top left 13 - top right 14 - bottom right 17 - bottom left
apple:zoom	The specified object scales up or down	animationType -- whether the object appears or disappears 1 - appear 2 - disappear direction -- whether the object scales up or down 1 - if object is appearing, it scales down to its final size; if object is disappearing, it scales up and disappears. 2 - if object is appearing, it scales up to its final size; if object is disappearing, it scales down and disappears
apple:spin	The specified object spins rapidly around its center while dissolving in or out	animationType -- whether the object dissolves in or out 1 - dissolve in 2 - dissolve out
apple:wipe	A cinematic wipe effect reveals or removes the object	animationType -- whether the object appears or disappears 1 - reveal 2 - remove direction -- the direction of the wipe 15 - left to right if revealing, right to left if removing 17 - right to left if revealing, left to right if removing 19 - top to bottom if revealing, bottom to top if removing 21 - bottom to top if revealing, top to bottom if removing

Back to top

Gradient and Image Shape Fills

Recall that previously we created a shape with a color fill, and we said that we could do two other kinds of fills with it -- gradient and image fills. Let's see how to do those now. The sixth slide in our sample document contains two shapes, a square and a circle. The square has a gradient fill, and the circle has an image fill. Here's the `shape` tag for the circle:

Listing 6. XML from Slide 6

```
<shape stroke-color="g1 0.5" path="M 85.3553 14.6447 C 104.882 34.1709
 104.882 65.8291 85.3553 85.3553 C 65.8291 104.882 34.1709 104.882 14.6447
 85.3553 C -4.88155 65.8291 -4.88155 34.1709 14.6447 14.6447 C 34.1709
 -4.88155 65.8291 -4.88155 85.3553 14.6447" opacity="0.9" stroke-width="1"
 transformation="3.285 0 0 3.28 346 220">
  <styles>
   <fill-style image-scale="scale-to-fill" image-data="tile_fun_purple.jpg"
    byte-size="27609" fill-type="image"/>
   <dash-style pattern="solid"/>
   <shadow-style opacity="0" radius="0"/>
  </styles>
</shape>
```

The only thing new about this object is that it has a different `fill-style` . The `fill-style` tag has a value of "image" for the `fill-type` attribute, and the other attributes specify information about this image. The `image-data` attribute and the `byte-size` attribute are just like attributes of the same name on the `image` tag of the previous slide. The `image-scale` attribute, however, is new. It describes what do with the image if it isn't exactly the same size as the shape it fills. In this case, we scale it so that it entirely fills the image. This attribute can take several values:

Table 5. Acceptable values of the "image" tag's "image-scale" attribute

Attribute value	Description
scale-to-fit	Scale the image so that it fits in the shape; if the image is larger than the shape, this scales it so that if can be contained inside the shape, while maintaining the aspect ratio. If the image is smaller, it scales it up.
scale-to-fill	Scale the image so that it fits in the shape, and fills the shape, possibly cropping the image, and maintaining the aspect ratio.
full-size	The image maintains its natural size. If it is smaller than the shape, then there is white space around it. If it is larger, then it is cropped.
stretch	If the image is smaller than the shape, stretches the image to fill the shape, changing the aspect ratio if necessary.

tile	If the image is smaller than the shape, then it displays the image multiple times to fill the shape.

The square has a gradient fill. The XML for the square looks like this:

Listing 7. XML from Slide 6

```
<shape path="M 0 0 L 100 0 L 100 100 L 0 100 Z M 0 0" transformation="1 0 0
 1 462 334">
  <styles>
    <fill-style fill-type="gradient">
     <gradient end-color="0.27451 0.27451 0.345098"
      start-color="0 0.504065 1" gradient-angle="270"/>
    </fill-style>
  </styles>
</shape>
```

As you can see, the only difference is the `fill-style` tag, which has a fill type of `gradient` . The gradient is expressed in the `gradient` child tag, which has three attributes. One, `start-color`, specifies the start color, and it is an RGB attribute, just like for single color fills. The attribute `end-color` is in the same format, specifies the end color. Finally, the attribute `gradient-angle` specifies the angle of the gradient, and like the `shadow` angle, can take any value from 0 up to, but not including, 360.

Back to top

Slide Fills

We can set the background of a slide in much the same way as we set the background of a shape. Consider slides 7, 8 and 9 of our sample presentation. Slide 7 has a gray background instead of a white one. The XML for that slide is as follows:

Listing 8. XML from Slide 7

```
<slide master-slide-id="master-slide-4">
  <drawables>
    <title visibility="tracks-master"
     vertical-alignment="tracks-master"/>
    <body visibility="tracks-master" vertical-alignment="tracks-master"/>
  </drawables>
  <transition-style type="inherited"/>
  <thumbnails>
```

```
        <thumbnail file="thumbs/st5.tiff" byte-size="504" size="60 45"/>
      </thumbnails>
      <bullets>
        <bullet marker-type="inherited" level="0">
          <content tab-stops="L 96" font-size="84" font-name="GillSans"
           paragraph-alignment="center"/>
        </bullet>
      </bullets>
      <background-fill-style fill-color="0.504065 0.504065 0.504065"
       fill-type="color"/>
    </slide>
```

Note that this isn't that much different from another blank slide -- the only addition is the `background-fill-style` tag, with the same attributes as the `fill-style` tag for a shape with a solid color fill.

Slide 8 has a gradient fill. Its `background-fill-style` tag looks like this:

Listing 9. XML from Slide 8

```
<background-fill-style fill-type="gradient">
  <gradient end-color="0.27451 0.27451 0.345098" start-color="1 1 1"
   gradient-angle="270"/>
</background-fill-style>
```

Again, we see that the `gradient` is described in the same way it would be for a shape.

Slide 9 has an image fill. Here is its `background-fill-style` tag:

Listing 10. XML from Slide 9

```
<background-fill-style image-scale="scale-to-fit"
 image-data="Aqua%20Blue.jpg" byte-size="264563" fill-type="image"/>
```

Again, we see similarity to the image fill style for a shape. In fact, all the attributes are the same, including the scaling.

Back to top

Bullet Basics

We've now got everything we need to create a fairly dynamic presentation: text, images, shapes, fills, transitions and builds. However, if you'll be working with a large amount of text, you won't want to want to deal with free-standing text labels. This is why Keynote provides bullet points.

On the 10th slide in our sample document, we start to see some bullet points. Let's look at the XML for that slide:

Listing 11. XML from Slide 10

```
<slide master-slide-id="master-slide-2">

  <drawables>

    <body visibility="tracks-master" vertical-alignment="tracks-master"/>

    <title visibility="tracks-master"

     vertical-alignment="tracks-master"/>

  </drawables>

  <transition-style type="inherited"/>

  <thumbnails>

    <thumbnail file="thumbs/st8.tiff" byte-size="814" size="60 45"/>

  </thumbnails>

  <bullets>

    <bullet marker-type="inherited" level="0">

      <content tab-stops="L 96" font-size="84" font-name="GillSans"

       paragraph-alignment="center">Bullet Points</content>

    </bullet>

    <bullet marker-type="inherited" level="1">

      <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"

       font-name="GillSans" paragraph-first-line-indent="7">Multiple

       levels</content>

    </bullet>

    <bullet marker-type="inherited" level="2">

      <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"

       font-name="GillSans" paragraph-first-line-indent="7">of

       points</content>

    </bullet>

    <bullet marker-type="inherited" level="3">

      <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"
```

```
           font-name="GillSans" paragraph-first-line-indent="7">are
           allowed</content>
        </bullet>
        <bullet marker-type="inherited" level="2">
          <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"
           font-name="GillSans" paragraph-first-line-indent="7">like
           so.</content>
        </bullet>
        <bullet marker-type="inherited" level="1">
          <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"
           font-name="GillSans"
           paragraph-first-line-indent="7">See?</content>
        </bullet>
      </bullets>
    </slide>
```

There are two differences between this and our previous slides. First, this slide uses a different master slide, "master-slide-2", which is a master slide that has both the title and body placeholders visible. Second, the `bullets` element is now more populated than on previous slides. We said before that every slide must have at least one bullet, but we didn't go into any detail about what that bullet looked like. Let's go into that detail now.

The first bullet on this slide has two attributes, `marker-type` and `level` . The "inherited" value on `marker-type` simply means that the glyph displayed next to the bullet is inherited from the master slide. The level, "0", means that this bullet is a top-level bullet. Top-level bullets are displayed in the title placeholder -- that "title" tag that has appeared in the `drawables` array of every slide. If the title placeholder isn't visible, then the top-level bullet won't be displayed.

Every slide has to have a top-level bullet (a bullet with a value of "0" for its `level` attribute), and it must always be the first bullet. You may have only one top-level bullet on a slide.

The `content` child of this bullet is just like the `content` child of the textbox in the textbox example. It specifies text and formatting for a bullet point. We'll discuss text formatting in more detail in a moment.

If you view the slide in Keynote, you'll see that the bullets have a hierarchical organization. However, this organization is not mirrored in the file format. Instead, it is derived from the `level` attribute on the bullet tags.

All of the bullet points that appear in the body placeholder have a `level` of "1" or more. The first point in a body placeholder always has a `level` of "1", and any siblings of that point also have a `level` of "1". Children of a `level` "1" point have `level` "2", and children of a `level` "2" point have `level` "3", and so on.

There are constraints on the level attribute that are not easily expressible in an XML schema or DTD. You may not "skip" a level -- in order to have a bullet point with `level` N + 2, there must be a preceding bullet point with `level` N + 1, and no preceding bullet point with a `level` less than M such that M is less than N + 1, unless there is also a point for every `level` between M + 1 and N + 1 inclusive. A couple examples might make this clearer:

Table 6. Examples of legal bullet structure

Legal Structure	Illegal	Reason for Illegality

	Structure	
Level 1 Level 2 Level 3 Level 4	Level 1 Level 2 Level 4	There isn't a point before the "Level 4" point that has a Level of 3.
Level 1 Level 2 Level 3 Level 2 Level 3 Level 4	Level 1 Level 2 Level 3 Level 2 Level 4	Although there is a point before "Level 4" that has a level of 3, the "level 4" point is also preceded by a point with level 2, and there isn't a level 3 point between them.

Back to top

Bullet Glyphs

On the 10th example slide, all of the bullets have the same glyph -- one inherited from their master slide. Usually inheriting from the master slide is the best way to go for providing a clean and consistent look, but sometimes you need something special. On slide 11, we present a variety of different bullet glyph styles. Let's consider the XML for those points one at a time.

The first point has the letter "A" for its bullet glyph. It is represented like this:

Listing 12. XML from Slide 11

```
<bullet marker-type="sequence" level="1">
  <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"
   font-name="GillSans" paragraph-first-line-indent="7">You can use
   sequences like letters...</content>
  <sequence-bullet-style format-string="%@." size-technique="relative"
   sequence-type="uppercase-alpha" size="1"/>
</bullet>
```

Note that the `marker-type` attribute is now "sequence" instead of "inherited". The exact appearance of this sequence is given by the `sequence-bullet-style` tag. The `size-technique` parameter states that the `size` attribute should be interpreted to mean that the size of this bullet varies with the text. (Keynote sets it to "1" for sequence bullets; you can set it to other values, but it might look odd.) The `format-string` parameter provides the format (e.g. "(A)", "(B)", "(C)"... or "--i--", "--ii--", "--iii--"), with "%@" standing for the value that's filled in (A, B, C..., i, ii, iii..., etc.) based on where in the sequence this bullet appears. Finally, the `sequence-type` attribute names the sequence, in this case "uppercase-alpha".

Note:
If you want a % sign to appear as part of a `format-string`, you need to use two % signs ("%%"). Also, if the "%@" sequence appears more than once in a `format-string`, Keynote may not load your document or may not display it properly.

This is the next bullet:

Listing 13. XML from Slide 11

```
<bullet marker-type="sequence" level="1">

  <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"

   font-name="GillSans" paragraph-first-line-indent="7">or

   numbers...</content>

  <sequence-bullet-style format-string="%@." size-technique="relative"

   sequence-type="arabic" size="1"/>

</bullet>
```

It's similar the last bullet point, but it has a different value for the `sequence-type` attribute -- "arabic", specifying Arabic numerals. Similarly the next bullet has a `sequence-type` of "uppercase-roman":

Listing 14. XML from Slide 10

```
<bullet marker-type="sequence" level="1">

  <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"

   font-name="GillSans" paragraph-first-line-indent="7">...or Roman

   numerals.</content>

  <sequence-bullet-style format-string="%@." size-technique="relative"

   sequence-type="uppercase-roman" size="1"/>

</bullet>
```

There are two other sequence types, "lowercase-alpha" (for a, b, c, etc.) and "lowercase-roman" (for i, ii, iii, etc.)

The next bullet does something different:

Listing 15. XML from Slide 11

```
<bullet marker-type="character" level="1" id="bullet-1">
```

```
    <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"
     font-name="GillSans" paragraph-first-line-indent="7">You can use any
     character.</content>
    <character-bullet-style offset="-13" size="72.0002">
      <bullet-characters tab-stops="L 84" font-size="11"
       font-color="System:textColor (0 0 0)"
       font-name="GillSans">*</bullet-characters>
    </character-bullet-style>
  </bullet>
```

It uses a character, in this case an asterisk, as the bullet point glyph. Note the different child element -- `character-bullet-style` in place of `sequence-bullet-style` . The text is encoded in much the same way as text for a bullet's contents, or a textbox.

Note also that this style has an attribute we haven't seen before, `offset` . This tells Keynote how many pixels to offset the bullet from the center line; since the asterisk character is drawn fairly high in this font, but we want it to appear centered, we've moved it down a few pixels. Also note that we do not specify a `size-technique` parameter, and as a result, the value of the `size` parameter is taken to be a literal value, not a multiplier.

Let's consider an image bullet next:

Listing 16. XML from Slide 11

```
<bullet marker-type="image" level="1">
  <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"
   font-name="GillSans" paragraph-first-line-indent="7">Image bullets are
   easy enough.</content>
  <image-bullet-style byte-size="19384" size-technique="relative"
   image-data="bullet_ball-chrome.tif" size="0.5"/>
</bullet>
```

Note that the `marker-type` attribute has a value of "image", and that we now have an `image-bullet-style` child tag. The `byte-size` and `image-data` parameters should be familiar -- they are just like the parameters in the `image` tag, or the `fill` for a shape or slide. The `size-technique` parameter has a value of "relative", which means that the image is scaled relative to the text -- in this case, to be half as high as the text. If the user changes the text size, the glyph changes size with it. If `size-technique` were absent, then the size would be interpreted as a scaling on the glyph itself, not in relationship to the text.

Finally, sometimes you want to make your point without a glyph, such as in the last bullet on this slide:

Listing 17. XML from Slide 11

```
<bullet marker-type="none" level="1">
  <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"
   font-name="GillSans" paragraph-first-line-indent="7">or you can skip the
   glyph entirely!</content>
</bullet>
```

Note that the `marker-type` attribute now has a value of "none".

Back to top

Building Bullets

We've seen how to apply builds to objects like images, shapes, and text boxes. We can do the same things to placeholders, which, as may recall, are the visual containers of bullet points.

Remember that `Level` "0" bullets (titles) are treated differently when they are displayed -- they appear in the title placeholder, instead of the body placeholder. Since builds are applied to placeholders, titles and other bullet points get built separately. Consider the `drawables` and `events` tags for the 12th slide of our sample document:

Listing 18. XML from Slide 12

```
<drawables>
  <body id="body-1" visibility="tracks-master"
   vertical-alignment="tracks-master"/>
  <title id="title-1" visibility="tracks-master"
   vertical-alignment="tracks-master"/>
</drawables>
<events>
  <build target-id="title-1" type="apple:move in" duration="1">
    <property name="direction" value="0"/>
    <property name="animationType" value="1"/>
  </build>
  <build target-id="body-1" type="apple:wipe" duration="1">
    <property name="direction" value="15"/>
    <property name="animationType" value="1"/>
  </build>
</events>
```

The first build applies to the object with the `id` of "title-1" -- which, as we can see from the `drawables` tag, is the title placeholder. It's just like the builds we've seen in previous examples.

Similarly, the second build applies to the object with the `id` of "body-1" -- the body placeholder. It also is just like all of the other builds we've seen so far.

Sometimes having all the bullets appear at the same time isn't what you want -- you want to pace the presentation of information to the audience. Keynote allows you to build bullets by group or point-by-point to facilitate this. Slide 13 has an example of building points group by group. The contents of the `drawables` tag are slightly different: since the body and title are different objects, you must specify different `id` attributes for them individually.

Listing 19. XML from Slide 13

```
<drawables>

  <body id="body-2" visibility="tracks-master"

   vertical-alignment="tracks-master"/>

  <title id="title-2" visibility="tracks-master"

   vertical-alignment="tracks-master"/>

</drawables>
```

The other difference on this slide is the contents of the `events` tag:

Listing 20. XML from Slide 13

```
<events>

  <build target-id="title-2" type="apple:move in" duration="1">

    <property name="direction" value="0"/>

    <property name="animationType" value="1"/>

  </build>

  <build target-id="body-2" duration="1" delivery="2" type="apple:spin">

    <property name="animationType" value="1"/>

  </build>

</events>
```

The `build` for the title placeholder is the same, but the body build has a new attribute, `delivery`, with a value of "2". Keynote interprets this to mean that it should build the bullet points one group at a time.

Slide 14 builds bullets one bullet point at a time. Its body placeholder has an `id` attribute of "body-3".

Listing 21. XML from Slide 14

```
<build target-id="body-3" duration="0.5" delivery="1" type="apple:zoom">

  <property name="direction" value="14"/>

  <property name="animationType" value="1"/>

</build>
```

We can see that it is identical to the `body` builds on the previous two slides, except that it has a value of "1" for the `delivery` attribute, which Keynote interprets to mean it should build the bullets one by one.

Back to top

Bullets Point Text

Slide 14 contains some highly formatted bullet points:

Listing 22. XML from Slide 15

```
<bullets>

<bullets>

  <bullet marker-type="inherited" level="0">

    <content tab-stops="L 96" font-size="84" font-name="GillSans"

     paragraph-alignment="center">Bullet

      <span font-name="GillSans-Italic">

        Text

      </span>

      is like Textbox

      <span font-name="GillSans-Italic">

        Text

      </span>

    </content>

  </bullet>

  <bullet spacing="10" level="1" marker-type="inherited">

    <content paragraph-head-indent="7" tab-stops="L 58" font-size="36"

     font-name="LucidaHandwriting-Italic" paragraph-first-line-indent="7">

      You can set the attributes for an entire point

    </content>

  </bullet>
```

```
    <bullet spacing="90" level="1" marker-type="inherited">
      <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"
       font-name="GillSans" paragraph-first-line-indent="7">
        You can
        <span font-name="GillSans-Italic">
          vary
        </span>
        the
        <span font-name="GillSans-Bold">
          text
        </span>
        <![CDATA[ ]]>
        <span font-name="LucidaHandwriting-Italic">
          within a
        </span>
        <![CDATA[ ]]>
        <span font-color="0 1 1" font-name="CopperplateGothic-Light">
          point
        </span>
      </content>
    </bullet>
    <bullet spacing="10" level="1" marker-type="inherited">
      <content paragraph-head-indent="7" tab-stops="L 58" font-size="42"
       font-name="GillSans" paragraph-first-line-indent="7">
        You have control over spacing between points.
      </content>
    </bullet>
  </bullets>
```

The text is formatted, you'll note, just like in the earlier textbox example. In fact, the `content` tag is identical to the `content` tag used in the textbox.

The only real difference between bullets and textboxes are that bullets have glyphs and they appear together in a placeholder. We've already covered glyphs. In order to provide more visual control, Keynote allows you to specify the space between bullet points. This is done with the `spacing` attribute on the `bullet` tag. We've done this here in an exaggerated manner to provide a visual demonstration. When you learn how Keynote describes themes and prototype bullet points, you'll see how this can be very useful for providing visual groupings of bullets.

Back to top

Conclusion

You've seen how Keynote represents slides, transitions, textboxes, shapes, images, bullets, and builds. In theory, you could take any of Keynote's themes and build an entire presentation just by filling out the `slide-list` element in your favorite text editor.

More usefully, though, you can now create applications that will create or change the contents of a Keynote presentation. This can result in very powerful tools for generating high-quality presentations automatically or more efficiently.

In Part II of this series, we'll describe how plugins -- including movies, charts and tables -- work, and how you can create and manipulate them.

Back to top

References

[1] Technical Note 2067: About the Keynote XML File Format (APXL Schema): http://developer.apple.com/technotes/tn2002/tn2067.html

[2] PictureShow (sample Java-based tool demonstrating APXL creation): http://developer.apple.com/samplecode/Sample_Code/Java/PictureShow.htm

[3] Open Darwin Keynote tools mailing list: http://www.opendarwin.org/mailman/listinfo/keynote-tools

[4] Bundles documentation: http://developer.apple.com/techpubs/macosx/Essentials/SystemOverview/Bundles/

[5] Scalable Vector Graphics (SVG): http://www.w3.org/TR/SVG/

[6] Basic Drawing: Coordinate Transforms (affine transformation matrix documentation): http://developer.apple.com/techpubs/macosx/Cocoa/TasksAndConcepts/ProgrammingTopics/DrawBasic/Concepts/transforms.html

Back to top

Downloadables

	Acrobat version of this Note (size in bytes, size in K)	Download
	TN2073 sample document.key	Download
	PictureShow (sample Java-based tool demonstrating APXL creation)	Download

Back to top

Technical Notes by Date | Number | Technology | Title
Developer Documentation | Technical Q&As | Development Kits | Sample Code