



- Welcome to InDesign/InCopy CS2 developer training session. Today we are going to cover advanced graphics topics.




Prerequisites

- **InDesign/InCopy CS1 plug-in development training**
 - Downloadable at http://partners.adobe.com/public/developer/indesign/index_training.html
 - Especially Session 10: Graphics

2005 Adobe Systems Incorporated. All Rights Reserved. 2




- In this session, we assume you already completed InDesign/InCopy plug-in development training for CS1, downloadable at http://partners.adobe.com/public/developer/indesign/index_training.html, especially session 10: Graphics, which covered basic concepts and architectures of InDesign graphics.




Objectives

- **An overview of advanced graphic topics**
 - Graphic page items
 - Color management
 - Advanced graphic attributes
 - Drawing framework
- **This training does NOT**
 - Obsolete CS1 graphics training
 - Replace programming guide-Graphic fundamentals, graphic solutions or other tech notes

2005 Adobe Systems Incorporated. All Rights Reserved.




- The objectives of this training is to provide an overview of advanced graphic topics, such as graphic page items, color management, advance graphic attributes and drawing.
- Please note that, this training does not obsolete CS1 graphics training, does not replace Graphics Fundamentals part of InDesign/InCopy programming guide or working with graphics solution document. You will need to consult these documents or browsable API reference in the CS2 SDK for details of these topics.



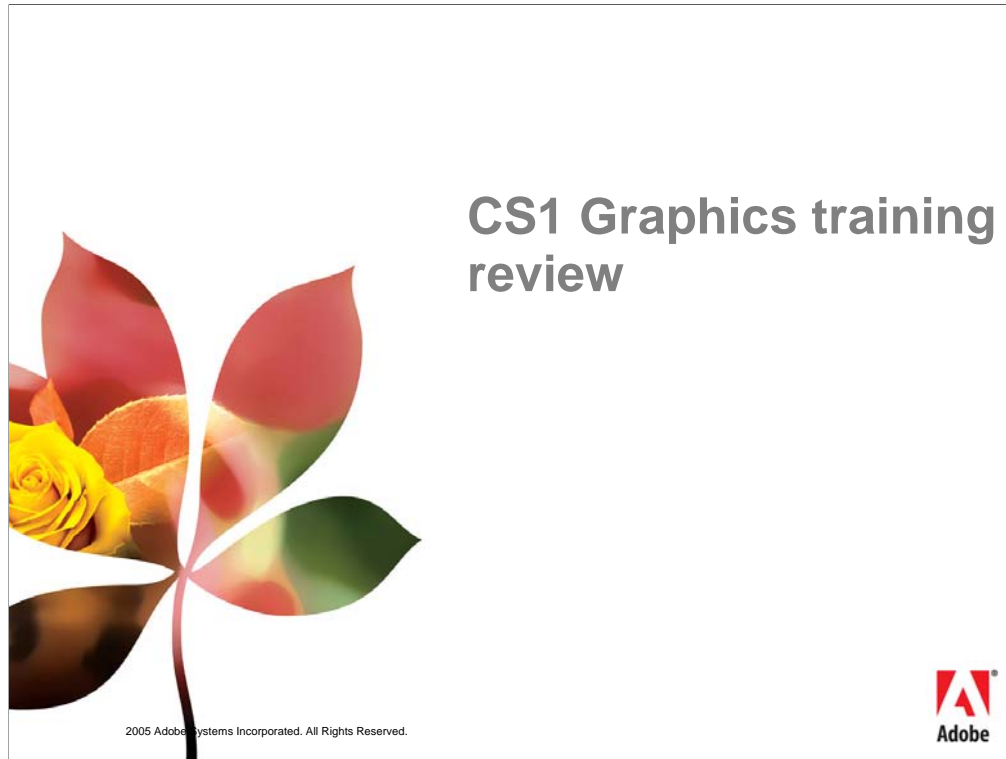
Outlines

- **CS1 graphics training review**
- **Graphic page items**
- **Color management**
- **Advanced graphic attributes**
- **Drawing framework**
- **Page item adornment**
- **Summary**


2005 Adobe Systems Incorporated. All Rights Reserved. 4



- In this session, we will first review CS1 graphics training contents, then we will cover new topics one by one.
- CS1 contents have covered how different graphic files are imported into InDesign. In this session, we will focus on the data model of graphic page items, and how we can export to various graphic file formats.
- Color management is very complex, we will cover only its main concepts and how it works in InDesign.
- We will also cover some advanced graphic attributes such as rendering attributes, stroke effects, and transparency.
- In drawing framework part, we will talk about the data structures for drawing, the drawing process and how to participate in drawing process by custom draw event handler.
- Page item adornment is somewhat related to page item drawing, but we listed here as a separate topic because it has its own architecture.




- Now let's take a look at what we have already covered in CS1 graphics training.



CS1 Graphics Review: Overview

- **Paths**
- **Graphic attributes**
- **Color model**
- **Working with graphic file formats**

2005 Adobe Systems Incorporated. All Rights Reserved. 6



- We had a brief description of the data model and use cases of the following topics: paths, graphic attributes, color model, and work with graphic file formats.



CS1 Graphics Review: Paths


- Splines are represented by `kSplineItemBoss`
 - `IPathGeometry` encapsulates path data
- Child of spline item(graphic content) may have other type of paths
- Manipulate spline items
- Utils and Suites
 - New in CS2: `IConvertShapeSuite`

2005 Adobe Systems Incorporated. All Rights Reserved.

7




- Let's first look at paths. Splines are the most common type of paths. They are represented by `kSplineItemBoss`. The examples of splines include paths drawn by pen tool, pencil tool, and frame tools such as rectangle, oval etc.
- The `IPathGeometry` interface encapsulates the path data. It stores all the path points that a page item may have.
- Spline items may have text or graphic contents. We call graphic contents as graphic page item, which may have clipping path and text wrap paths. we will discuss them more later.
- We had talked about manipulating spline items, such as creating, deleting spline item, or modifying paths using various commands, utils, and suites. In CS2, we also introduced a new suite called `IConvertShapeSuite` that can convert selected spline items to different shapes.



CS1 Graphics Review: Graphic attributes

- Represented by graphic attribute bosses
 - Aggregating IGraphicAttributeInfo
- Master list of graphic attributes
- Applied attributes are represented as a list of graphic attribute bosses
- Defaults are stored in kGraphicStateBoss
- Graphic attribute values
- Utils and suites

2005 Adobe Systems Incorporated. All Rights Reserved. 8



- A graphic attribute describes a particular property of how an page item should look on a layout. It is represented as a graphic attribute boss, inheriting from kGraphicsAttrBoss and aggregating IGraphicAttributeInfo.
- We provided a master list of graphic attributes in CS1 graphics training session. CS2 only add 4 additional attributes related to transparency. You can search for kGraphicsAttrBoss, or IGraphicAttributeInfo in the reference documentation to see the attributes.)
- Graphic attributes applied to page items are represented as a list of graphics attribute bosses. You can get applied attributes from IGraphicStyleDescriptor. You may also apply graphic attributes using several utility interfaces and suites. Please refer to Solution document “working with graphics” for examples.
- Default graphic attributes are stored in kGraphicStateBoss. When you create new page items, these attributes in the graphic state boss are copied and applied to new page items if you specify INewPageItemCmdData::kDefaultGraphicAttributes in spline creation methods on IPathUtils (see attrType parameter).
- Some graphic attributes have simple values, such as int32, boolean etc. Some graphic attributes have more complicated values, such as using UID to reference an object, using boss Class ID to reference an implementation, etc. We will see examples later.



CS1 Graphics Review: Color


- Swatch, the rendering object
- Color space and components
- Color tint, gradient, mixed ink, and swatch list
- Ink
 - IPMInkBossData
- Ink list
- Use cases
 - Get color info,
 - Create swatches etc.

2005 Adobe Systems Incorporated. All Rights Reserved.

9




- We had talked about color model. Swatches are rendering objects, they could be solid color, tint, gradient, mixed ink etc. They are represented by `KPMColorBoss`, `kGradientRenderingObjectBoss`, `kGraphicStateNoneRenderingObjectBoss`, `kAGMBlackBoxRenderingObjectBoss` etc.
- Color can be defined as within a color space with numerical values for each component. For example, in CMYK, each of C, M, Y, K component could be real values ranging from 0 to 1.
- Colors can be broken down to inks. The boss that represents an ink is `kPMInkDataBoss`.
- Ink list holds the inks used. It is represented by `lInkList`, aggregated on `kWorkspaceBoss`, `kDocWorkspaceBoss` and `kBookBoss`.
- We also talked about some use cases, such as getting information of a color, creating color or gradient swatches etc. Please refer to solution document “working with graphics” for examples.



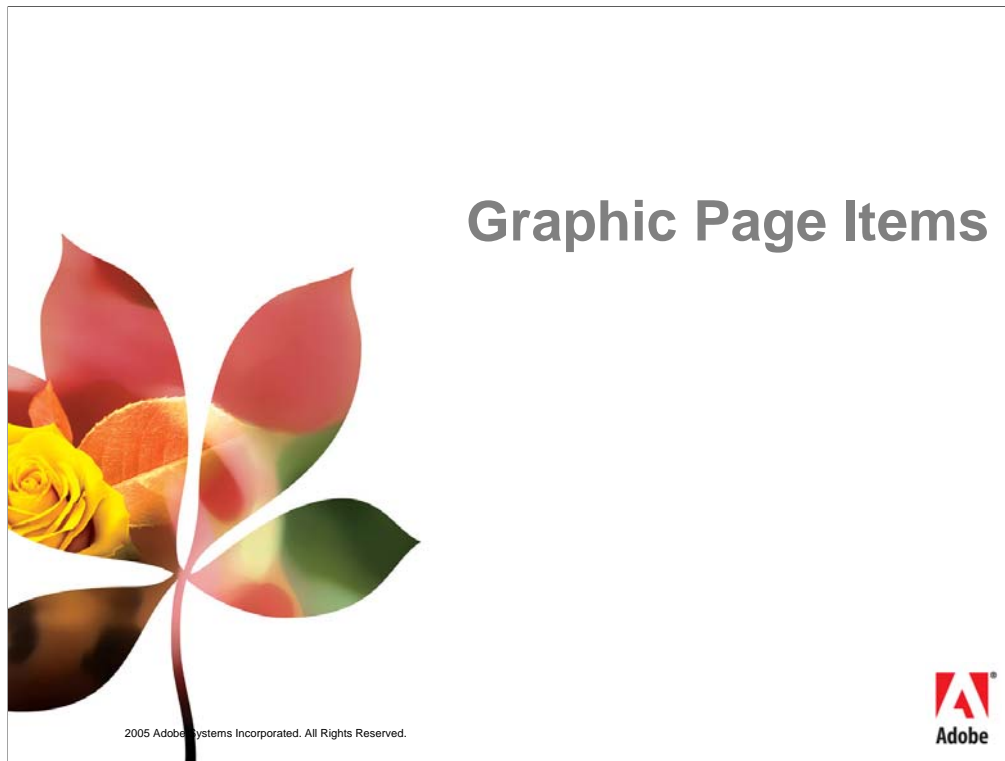
CS1 Graphics Review: Graphic file format

- Graphic file formats
 - Vector graphics and raster images
- Content item boss for each file formats
- Modify import options


2005 Adobe Systems Incorporated. All Rights Reserved. 10



- We talked about graphic file format InDesign supports. InDesign supports importing most of vector graphics and raster image files. Each of the file formats corresponds to a graphic page item boss (for example, kImageItem), usually created as a child of kSplineItemBoss during import.
- Import options for different file format varies. Import options are stored as application preferences and you can modify import options by executing their respective commands, such as kSetEPSPrefsCmdBoss, kSetPDFPlacePrefCmdBoss.




- Now lets get started with new contents, the first is Graphic page items.



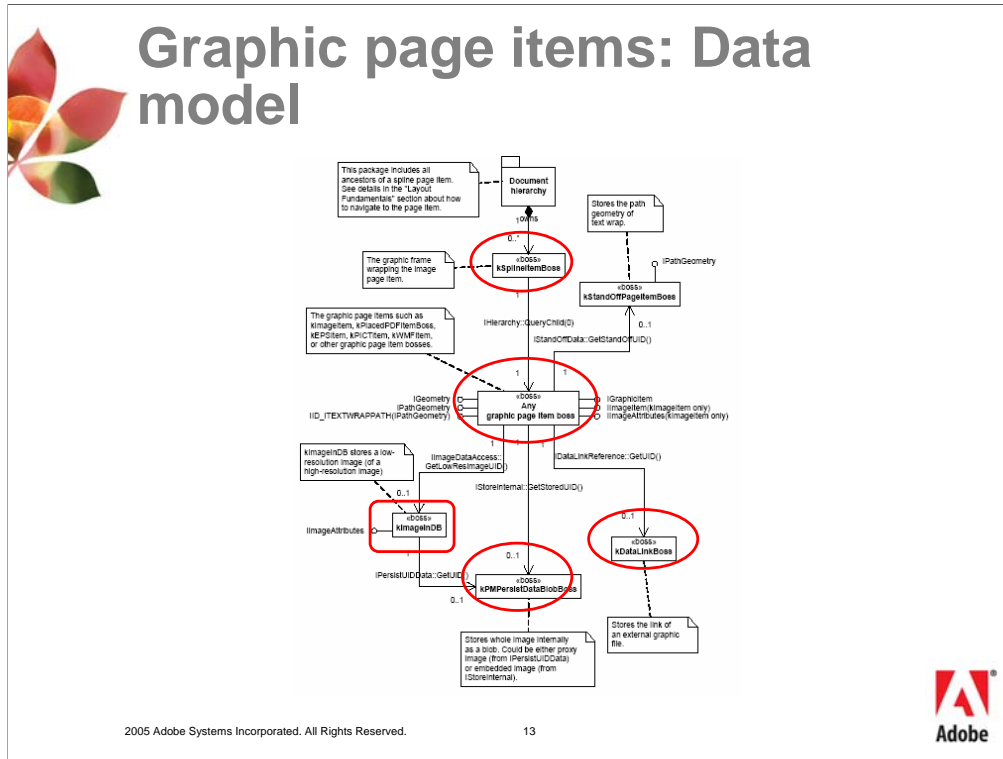
Graphic page items: Overview

- **Graphic page item data model.**
 - Proxy Image
- **Graphic page item settings**
 - Clipping path
 - Text wrap
 - Display performance
- **Export to Graphic file format**
 - SVG, JPEG


2005 Adobe Systems Incorporated. All Rights Reserved.



- We just reviewed that each file format has its corresponding graphic page item. Now we will focus on the data model of graphic page item. We will introduce the proxy image concept and explain how proxy image is used. We will also talk about graphic page item's special settings, namely clipping path, text wrap and display performance. We will also briefly discuss exporting to various graphic file formats.




- Let's look at the graphic page item's data model. This structural diagram can be found in Graphics Fundamentals part in CS2 Programming guide. From the document hierarchy, **<Click>** we will get the spline item first, we call this the graphic frame. The graphic page item is the graphic frames' only children, **<Click>** which can be any type of graphic page items, such as kPlacePDFItemBoss for PDF, KEPSItem for EPS or kImageItem for various raster image formats.
- Some of the interesting interfaces are listed, for example, IPathGeometry on any graphic page item boss stores path information of the graphic page item's clipping path. We also have IDataLinkReference interface referring to a kDataLinkBoss **<Click>** to store the link to the graphic file where the graphic page item was originally imported from. When the image file is embedded **<Click>**, we have IStoreInternal interface pointing to data blob that stores image data.
- Also, **<Click>** during graphic file import, we created a proxy image that is suitable for display at the screen for a better performance. It is accessible through ImageDataAccess.




Graphic page item: Data model

- **Proxy Image**
 - Low-resolution image for viewing
 - Does not affect printing
 - Represented by `kImageInDB`
 - Stores image data in `kMPersistDataBlobBoss`

2005 Adobe Systems Incorporated. All Rights Reserved. 14




- Proxy image is a low-resolution image that is optimized for viewing on screen. You can scroll, resize and transform the proxy image without reading original high-resolution image file, but still get a sense how the image will look like. One thing I want to point out is that it is for viewing purpose only, and has no effect on printing— We always use the original file when the document is sent to print or exported to PDF.
- The proxy image is represented by `kImageInDB` boss. The `IPersistUIDData` on this boss points to an `kMPersistDataBlobBoss` object, which stores proxy image data as blob.




Graphic page item: settings

- **Clipping path**
 - Crops the image
 - Stores path in IPathGeometry of graphic page item (separately from graphic frame)
 - IClipSettings
 - IClippingPathSuite

2005 Adobe Systems Incorporated. All Rights Reserved. 15




- A clipping path crops part of the image so that only a portion of the image is drawn. The clipping path is stored in IPathGeometry of the graphic page item, separately from its graphics frame. So you can freely modify one without affecting the other. Note that the clipping path does not change the image, it only affects how the image is drawn.
- IClipSettings interface is aggregated on kWorkspaceBoss, kDocWorkspaceBoss, and every graphic page item boss to define application, document, or individual graphic page items' clipping path settings preferences. You can set clipping path to the alpha channel or Photoshop paths embedded in the image, or you can let InDesign detect edges of the graphics automatically. You may also draw your own path or modify the IPathGeometry on the graphic page item directly using appropriate commands.
- We recommend you use IClippingPathSuite whenever possible to set clipping path. For usages, please refer to "Working with graphics" and code snippet "SnpManipulatePathandGraphics".




Graphic page item: settings

- **Text wrap**
 - Could apply to any page item
 - Text wrap mode
 - Stores path as IPathGeometry on kStandOffPageItemBoss
 - Text wrap contour option is for graphic page item only
 - IStandOffContourWrapSettings
 - ITextWrapFacade

2005 Adobe Systems Incorporated. All Rights Reserved. 16




- Text wrap defines how text is composed around an object. You can apply text wrap to frame of any page item by setting appropriate text wrap mode.
- Text wrap path is stored in IPathGeometry interface on a separated kStandOffPageItemBoss object, referenced through IStandOffData interface aggregated on both graphic frame and graphic page item bosses.
- What makes graphic page item special is that when the text wrap mode is set to “Wrap around object shape”, (i.e., IStandOff::kManualContour), we are able to set text wrap contour options. The text wrap contour options is stored in IStandOffContourWrapSetting. Very similar to clipping path, The interface is aggregated on kWorkspaceBoss, kDocWorkspaceBoss, and every graphic page item bosses to define contour settings preferences in different levels. You can set contour setting to the graphics’ height and width, or other settings that a clipping path has. You can even set it directly as “Same as clipping”.
- We recommend using ITextWrapFacade to access or set text wrap mode and text wrap contour options. For example, please see “work with graphics” and snippet code SnpManipulatePathandGraphics.




Graphic page item: settings

- **Display performance**
 - For performance in the screen drawing
 - Groups:
 - Fast, typical, high quality
 - See also proxy image
 - 3 levels of settings
 - Session: IDrawOptions on kWorkspaceBoss
 - View: IDrawOptionsSetID on kLayoutWidgetBoss
 - Object: IDrawOptionOverrides on kDrawablePageItemBoss
 - IDisplayPerformanceSuite

2005 Adobe Systems Incorporated. All Rights Reserved. 17




- Display performance is used to set the balance between graphic display speed and quality. You can specify the quality of how raster images and vector graphics are drawn to the screen.
- A display performance group is a set of values for these categories. There are 3 groups available:
 - Fast: The highest display speed, but the lowest display quality. In the standard setting for this group, InDesign draws a raster image or vector graphic as a gray box.
 - Typical: The default option for graphic page items. In the standard setting for this group, InDesign draws a low-resolution proxy image appropriate for identifying and positioning a graphic. I hope you all remembered that the “proxy image” is generated during import process.
 - High Quality: The highest quality but the lowest display speed. In the standard setting for this group, InDesign draws a raster image or vector graphic at high resolution (i.e., uses the file the graphic page item’s datalink provided).
- Note these standard setting is just the defaults that follows the group name, there is no restriction on what you can set, for example, you can even set high quality group to display gray box for raster images.
- There are 3 levels of display performance settings.
 - At the session level, IDrawOptions is aggregated on kWorkspaceBoss and stores the default display performance setting. The display performance groups are defined in this interface and can be accessed by other two levels by a GroupID.
 - At layout view level, IDrawOptionsSetID aggregated on kLayoutWidgetBoss stores view display performance preferences.
 - And at object level, IDrawOptionOverride interface aggregated on kDrawablePageItemBoss stores individual object overrides.
- We recommend using IDisplayPerformanceSuite to set selected object’s display performance. See “working with graphics” and SnpManipulateDisplayPerformance for examples of change display performance group settings and others.



Graphic page item: Export

- **Export to graphic file formats**
 - PDF, EPS, SVG, JPEG
- **Export process**
 - Drawing page item to intermediate entity
 - Writing to file stream
- **Export options**
 - Varies with different format
 - Can be set via commands


2005 Adobe Systems Incorporated. All Rights Reserved. 18



- InDesign supports exporting all or part of a document to PDF, EPS, SVG, and JPEG formats. They are implemented using service provider mechanism.
- The general export process usually involves in two steps. First is to draw content to an intermediate entity, such as a view port; the second step is to write content to the provided file stream.
- Export options are session preferences and vary with different file formats, they can be set via appropriate commands.
- Exporting PDF and EPS have already been discussed in detail in CS1 Training session 15 Printing.
- Please see details on SVG and JPEG exporting in Graphic Fundamentals.




- Now we will discuss color management mechanism of InDesign CS2



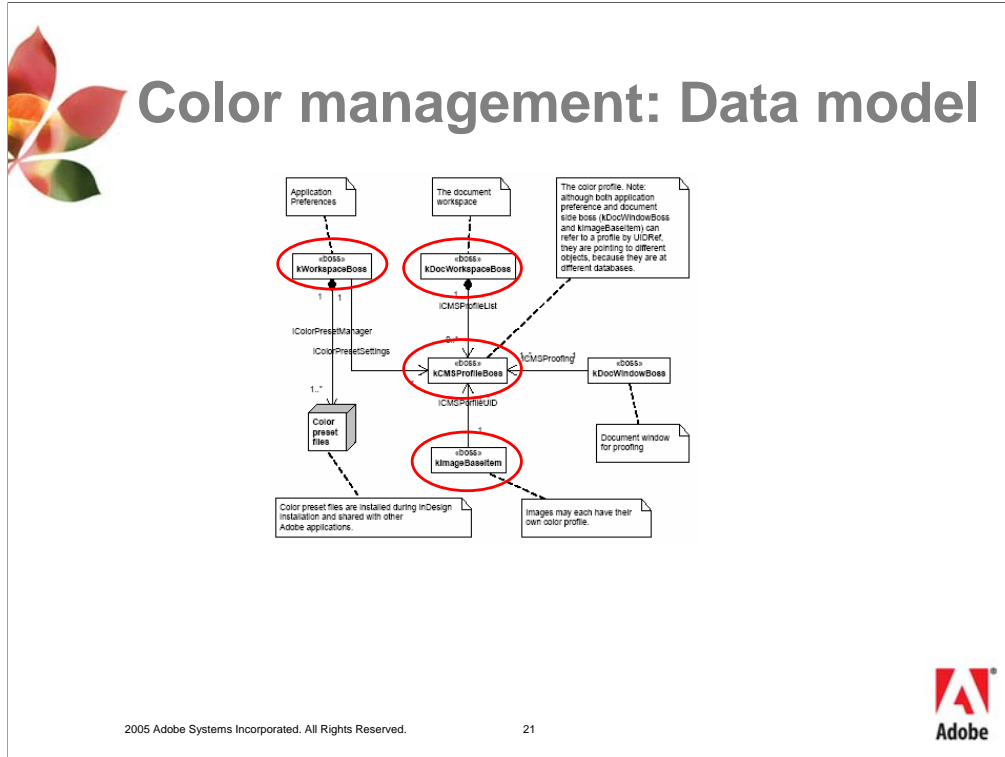
Color management: Basic

- **ICC Profiles**
 - Is a record of unique color characteristics of input/output device
 - Contains data for translation of device-dependent color to L*a*b* color
- **InDesign color management**
 - Supports multiple color spaces in one document
 - Works with other application to produce consistent color
 - See InDesign help


2005 Adobe Systems Incorporated. All Rights Reserved. 20



- Color management is complex and has a lot of theories behind it. We will only talk about some basics.
- There are three commonly used color spaces: RGB, CMYK, and L*a*b*. L*a*b* is a device independent color space that has its basis in the CIE standard observer.
- The objective of color management is to keep consistent color across applications, platforms, and different devices. ICC profile is introduced to achieve this goal. ICC profile is a record of the unique color characteristics of a color input or output device, and can provide translations of device-dependent color to L*a*b* color. Thus in order to translate color from one device to another, we only need to translate to Lab first, then translate from Lab to destination device.
- InDesign color management is complicated because it supports multiple color space in one document. For example, even a document is set to use RGB, an imported image can still use CMYK.




- This figure illustrates the InDesign color management data model, **<Click>** kCMSProfileBoss represents ICC color Profile. **<Click>** ICMSProfileList, aggregated on document workspace boss, stores all color profiles used in the document.
- **<Click>**the session preference kWorkspaceBoss aggregate (1) IColorPresetManager that manage color presets, and (2) IColorPresetSetting that store working color management settings. **<click>** Imported images may have their own profiles.




Color management: Data model

- **Color presets**
- **Color Profile**
 - Represented by `kCMSProfileBoss`
- **Image color setting**
 - `ICMSItemProfileSource` aggregated on image boss

2005 Adobe Systems Incorporated. All Rights Reserved. 22




- Color presets are a set of files that store predefined color management settings, and are saved on hard disk during Adobe Creative Suite 2 installation. They are shared among different adobe applications to achieve consistent color across the suite.
- As shown in previous slide, color profiles are represented by `kCMSProfileBoss`. The `ICMSProfile` stores type and category of the profile and can get/set profile data by accessing `IACESpecificProfileData` interface on the same `kCMSProfileBoss`.
- Any imported image can either have an embedded profile or have a profile already assigned to it. (See Object > Image Color Settings). The interface `ICMSItemProfileSource` aggregated on the boss class `kImageBaseItem` stores the basic profile assignment type (see the enumeration `ICMSItemProfileSource::ProfileSourceType`) and the name of the image's embedded profile, if any.



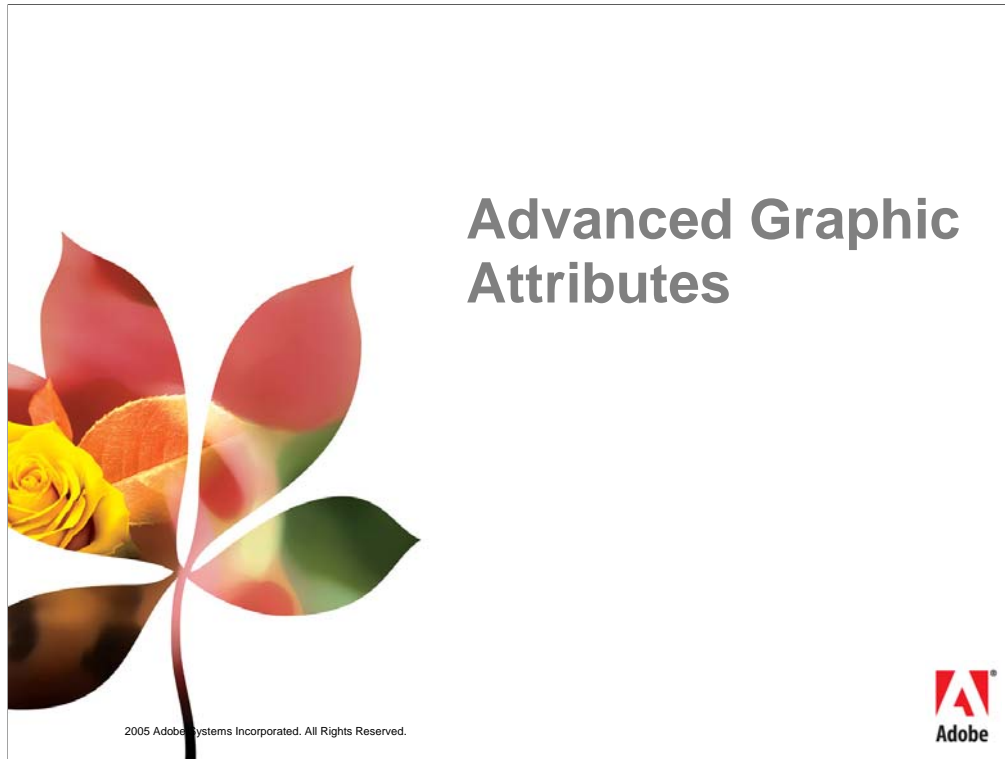
Color management: use cases

- Get linked image's ICC profile
 - ICMSItemProfileSource on kImageBaseItem for profile type. If embedded, then
 - Get UID from
IPersistUIDData(IID_ICMSPROFILEUID)
- Key utils and interfaces
 - ICMSUtils
 - ICMSManager
 - IColorPresetsManger
 - ICMSProfileList


2005 Adobe Systems Incorporated. All Rights Reserved. 23



- Use case: get a linked image's ICC Profile. If an image uses an external profile or an embedded profile, then the interface IPersistUIDData(with interface identifier IID_ICMSPROFILEUID) on kImageBaseItem stores the UID of an instance of kCMSPProfileBoss, so you will be able to instantiate any interface on kCMSPProfileBoss from this UID and get the profile data you want.
- There are several key utility and manager classes used to access color management system. "Working with graphics" solution document has examples of turning on/off color management system, and other color management related questions and answers.




- In CS1 graphics training session, we've already talked about graphic attributes in general. Now lets look at some advanced graphic attributes.



Adv. graphic attributes: Overview

- **Rendering attributes**
 - Color rendering attributes
 - Gradient rendering attributes
- **Stroke effects**
 - Path stroker
 - Path corner
 - Path end stroker
- **Transparency**
 - Transparency concepts
 - Data model

2005 Adobe Systems Incorporated. All Rights Reserved. 25



- We will talk about how to render a page item first, including colors and gradient as rendering attributes.
- Then we will cover how to customize your own stroke effects.
- We will also discuss transparency concepts and how to achieve the transparency effects.



Adv. graphic attributes: Rendering attributes


- **Color as an graphic attribute vs. color as an object**
- **Attribute whose value are UIDs of swatches**
- **Color rendering attributes**
 - Stroke, Fill and Gap are rendered independently.
- **Gradient rendering attributes**
 - Multiple attributes work together to define a gradient (angle, center, length, radius etc.)
 - `kGraphicStyleGradient<something>AttrBoss` in master attribute list

2005 Adobe Systems Incorporated. All Rights Reserved.

26




- We have talked about color as object in CS1 graphics training. We know how colors and gradients are defined. However, in order to let InDesign draw a page item in a particular color or gradient, you must have a way to tell it. This come the colors and gradients as rendering attributes.
- These rendering attributes have values whose type are UIDs. These UID points to the swatches such as color and gradient, so that InDesign knows to use the referred swatch to draw page items.
- You can always render a page item's stroke, fill and gap in different color or gradient independently. For example, you could set stroke color to be black, but fill color to be red.
- If you want draw page items using gradient, you will need multiple attributes works together. You will need to provide angle, center, length, and radius (if its type is radius). There are a bunch of attributes named in form of `kGraphicStyleGradient<something>AttrBoss` in the master attribute list that are used to describe gradient rendering.



Adv. graphic attributes: Stroke effects

- **Path stroker**
 - kGraphicStyleStrokeLineImplAttrBoss stores ClassID of stroker implementation
 - IPathStroker
 - IPathStrokeList
- **Path corner**
 - kGraphicStyleCornerImplAttrBoss stores ClassID of corner implementation
 - IPathCorner
- **Path end (arrow head)**
 - kGraphicStyleLineEndStartAttrBoss and kGraphicStyleLineEndEndAttrBoss stores ClassID of path end implementation
 - IPathEndStroker

2005 Adobe Systems Incorporated. All Rights Reserved. 27



- Stroke effects are those attributes that can choose a stroker implementation to display page items. The values of these attributes are ClassIDs of implementations. In fact, these ClassIDs are the service provider of stroker services.
- The first we will discuss is the path stroker. InDesign has several predefined stroker implementation accessible through IPathStrokerList aggregated on kWorkspaceBoss and kDocWorkspaceBoss. Path stokers have signature interface IPathStroker, and are referred by graphic attribute kGraphicStyleStrokeLineImplAttrBoss.
- Path corners have signature interface IPathCorner and are referred by kGraphicStyleCornerImplAttrBoss
- Path ends, also called arrow heads, have signature interface IPathEndStroker. Path ends actually have two related attributes kGraphicStyleLineEndStartAttrBoss and kGraphicStyleLineEndEndAttrBoss, corresponding to the start and end of a line or curve.
- Note path corner and path end does not have interface similar to IPathStrokerList.



Adv. graphic attributes: Stroke effects (continued)


- **Customize path stroker**
 - provide kPathStrokerService
 - Implement IPathStroker
- **Customize path corner**
 - provide kPathCornerService
 - Implement IPathCorner
- **Customize path end**
 - provide kPathEndStrokerService
 - Implement IPathEndStroker
 - Stroke bounding box include both path end and path

2005 Adobe Systems Incorporated. All Rights Reserved.

28




- Customizing stroke effects is straightforward. Since they are implemented as service provider mechanism, you only need to provide specific service.
- For path stroker, provide kPathStrokerService and implement IPathStroker. For path corner, provide kPathCornerService and implement IPathCorner. For path end, provide kPathEndStrokerService and implement IPathEndStroker. Note for path end, the page item's stroke bounding box includes both path and the path ends, so you may need to adjust original path points to get the same bounding box with or without path ends.
- You need to set specific graphic attributes to the customized stroke effect ClassID to let the InDesign draw page item with the customized stroke effects.




Adv. graphic attributes: Transparency

- **Concepts**
 - Opacity and blend mode
 - Basic transparency
 - Drop shadow
 - Feather
 - Flattening

2005 Adobe Systems Incorporated. All Rights Reserved. 29




- Now let's introduce some transparency concepts. Opacity is the converse of transparency and is expressed as a percentage, where 0% relates to completely clear, and 100% relates to completely opaque. Page items can have individual opacities assigned to them. Blending mode defines how the background (backdrop) and foreground (source) colors interact. (they are analytic expressions that produce interesting visual effects.)
- Basic transparency is an effect that lets the viewer see through an object. End-users can specify the transparency attributes of selected page items through Transparency panel.
- Drop shadow is a blurred representation of the shape of an object, offset by a user-specified distance and drawn below the object. An end-user can set a drop shadow through the Drop Shadow dialog box.
- Feather is a graphic effect that allows designers to create a smooth edge around a frame. The UI for creating feather effects is the Feather dialog box.
- Some device do not have transparency representation, such as printer. Flattening is the process of generating nontransparent representations of the transparency effects.



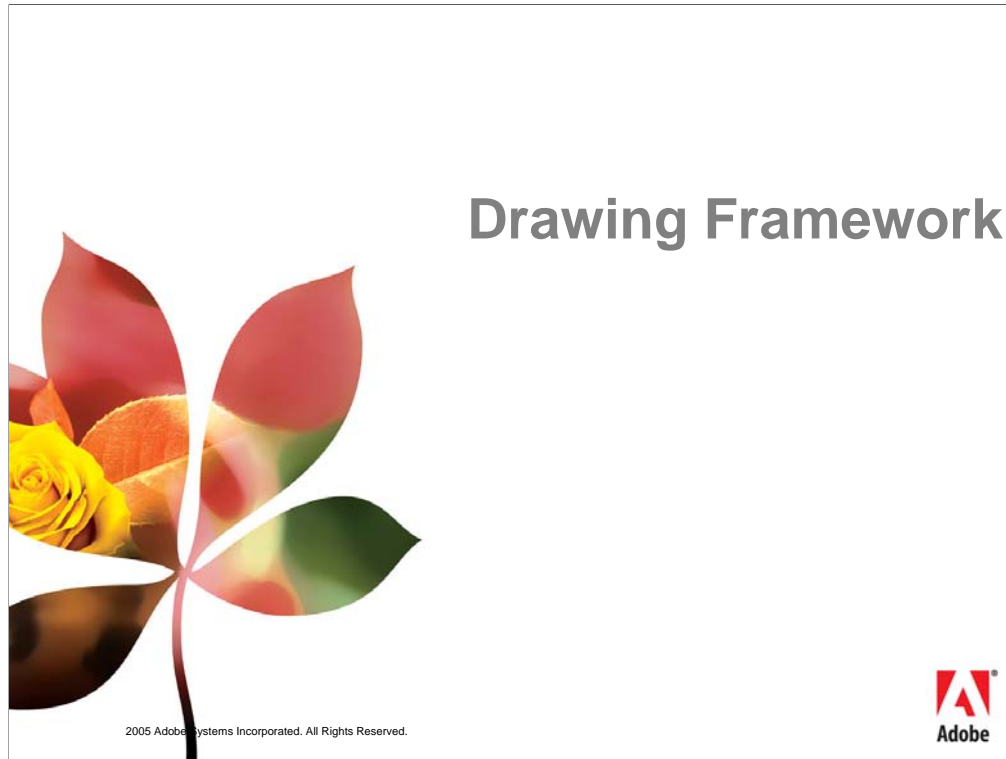
Adv. Graphic attributes: Transparency

- **Data model**
 - Define transparency as graphic attributes
 - Draw transparency using as page item adornment
 - Will be covered later.
 - Flattening transparency during print


2005 Adobe Systems Incorporated. All Rights Reserved. 30



- Transparency feature is complex, but the principle to implement transparency is not.
- Transparency is defined by graphic attributes. These attributes specifies parameters of the transparency effects. For example, to define basic transparency, you have to specify opacity and blending mode. There are a bunch of attributes starting with kXP in the graphic attribute master list work together to define transparency effects.
- Transparency is drawn using page item adornment mechanism, which we will cover later in this session.
- We flattens transparency during print.
- For examples of implementing new transparency enhancement, see TransparencyEffect and TransparencyEffectUI sample plugin in SDK.




- Now we will discuss InDesign drawing mechanism




Drawing framework: Overview

- **Data structures for drawing**
 - Windows
 - Graphics context
 - View port
- **Drawing dynamics**
 - Drawing the layout
 - Drawing page items
 - Drawing in UI widget windows
 - Offscreen drawing
- **Draw event handler**
 - Custom draw event handler

2005 Adobe Systems Incorporated. All Rights Reserved.




- We will cover basic data model for drawing first, then the dynamics of drawing, namely how we draw the document layout, page items, in UI widgets and on offscreen.
- We will also talk about how to participate in drawing process by adding custom draw event handler.




Drawing framework : Data structure

- **Windows**
 - Layout windows
 - UI Windows
- **Graphics context**
 - Encapsulate data for device-independent drawing
 - Derived from IGraphicsContext
 - Can be instantiated with
 - A view port
 - A view (IControlView)
 - An update region
- **View port**
 - Platform independent drawing API
 - Draws through AGM (Adobe Graphics Manager)

2005 Adobe Systems Incorporated. All Rights Reserved. 33




- Let's look at some basic constructs of drawing first.
- Windows are the targets of drawing. Drawing documents to the screen really means drawing to an application layout window. In addition to the layout window, there are various other windows. Dialog boxes, panels, and even pop-up tips are all application windows. We call these UI Windows.
- Windows are represented by a platform-independent kWindowBoss object. Its interfaces provide basic window manipulations such as move, resize, etc.
- Graphics context encapsulates data necessary for device-independent drawing. It is derived from IGraphicsContext, an abstract data container interface. A graphic context can be instantiated with a view port, a control view and an update region.
- Viewports are bosses that provide platform independent drawing API. Actual drawing is delegated to AGM (Adobe Graphics Manager).
- Not only screen drawing has a view port, in fact, PDF and SVG exporting also draws to their own export view ports.




Drawing framework : Dynamics

- **Layout drawing**
 - Invalidating a view
 - Invalidation state maintained by OS
 - ILayoutUtils::InvalidateViews
 - Drawing order
 - Background and foreground
 - Z-order (Windows Z-order, Layers, Page item Z-order)
 - IDrawManager
 - On each view port
 - Provide drawing service to any page element

2005 Adobe Systems Incorporated. All Rights Reserved. 34




- Layout drawing is to draw document contents to the layout window. Operating systems are responsible to redraw the window thus they maintain invalidation states of the windows. You can explicitly tell OS to invalidate the whole window by calling ILayoutUtils::InvalidateViews method .
- The layout window maintains background and foreground offscreen, which we will discuss later. The background refers all these page item that the document has, the foreground refers the selection handles.
- InDesign honors three levels of z-order: Window z-order, Layer order and page item Z-order. At window-level, InDesign draws the furthest window first, then one by one until the nearest one. At layer level, InDesign draws page layer first, then other document layers. On the same layer, page items are drawn according to their positions at the parent object's hierarchy. The child with index 0 is drawn first; the child with the greatest index is drawn last.
- IDrawManager, an interface exists on every view port, is used to set clipping areas and initiate the drawing of any page element and its children to the view port. The draw manager provides hit testing, iterating drawing order, and drawing service to each page item, and provide a mechanism to interrupt the drawing process at page item level.




Drawing framework : Dynamics

- **Page item drawing**
 - Drawing context
 - GraphicsData: the wrapper of IGraphicsContext
 - Drawing flag
 - IGraphicsPort
 - Aggregated on view port for drawing
 - gsave(), grestore() pair
 - IShape: drawing, hit testing etc.
 - IHandleShape: drawing, hit testing for selection.

2005 Adobe Systems Incorporated. All Rights Reserved. 35




- Now let's look at how page items are drawn.
- Page item boss aggregates IShape interface, which is responsible for drawing and hit testing. In the IShape::Draw() method, a GraphicsData type parameter and an integer parameter representing drawing flag are passed in. The GraphicsData is the wrapper of IGraphicContext.
- IGraphicPort interface is aggregated on view port boss for drawing. Its methods are very similar to PostScript graphics operators. GraphicPort's methods support drawing primitives, port transformation and manipulation, and changing the port clip settings.
- IGraphiPort's methods gsave() and grestore() pair are used to effectively push the current port settings onto a stack and pop the old settings when desired. For example, when a page item is called to draw, it should save the port settings, set the port to its inner coordinate space, draw, and then restore the port before returning.
- Similar to IShape, IHandleShape is aggregated on page item bosses. It is used to draw page item selection handles.




Drawing framework : Dynamics

- **Page item drawing process**
 - 1) Broadcast pre-draw messages**
 - `kAbortCheckMessage`, `kFilterCheckMessage`, and `kDrawShapeMessage`
 - 2) Gsave()**
 - 3) Broadcast `kBeginShapeMessage`**
 - 4) Draw `kBeforeShape` page item adornments**
 - 5) Draw page item's own shape**
 - 6) Draw `kAfterShape` page item adornments**
 - 7) Grestore()**

2005 Adobe Systems Incorporated. All Rights Reserved. 36




- Although each IShape implementation may have its own specific drawing sequence, all implementations follow these steps as a guideline:
 - 1. When a page item begins drawing, three drawing events are broadcast (`kAbortCheckMessage`, `kFilterCheckMessage`, and `kDrawShapeMessage`). No IShape drawing activities occur between the broadcasts.
 - 2. Save the graphics port state by calling `IGraphicsPort::gsave()`. The graphics port should be set to draw in the page item's inner coordinate system.
 - 3. A `kBeginShapeMessage` drawing event is broadcast. If any drawing event handler returns `kTrue`, the drawing activity ends.
 - 4. Draw `kBeforeShape` page item adornments.
 - 5. Draw the page item's own shape by calling the protected method `DrawShape()`. This method varies depending on the nature of the page item. The following is a list of common tasks the `DrawShape` method may accomplish: Defining the item's path, filling the path, setting the port to clip to the path, drawing the item's children, stroking the path, etc.
 - 6. Draw `kAfterShape` page item adornments.
 - 7. Then restore the graphic port state.




Drawing framework : Dynamics

- **UI drawing**
 - Drawing context
 - IViewPort:
 - Update region
 - IControlView
 - Drawing directly within IControlView
 - Construct AGM directly
 - UI Color
 - GetFrame() returns position in parent coordinates

2005 Adobe Systems Incorporated. All Rights Reserved. 37




- UI Drawing refers to draw lines, curves and text directly in UI windows, such as dialog, palette, or widgets etc.
- The UI drawing code resides in IControlView::Draw(). It gets IViewPort and an update region as its drawing context. You can draw anything directly within IControlView.
- Normally you will construct an AGM object, then get IGraphicsPort from AGM, and begin drawing (See example PnlTrvCustomView.cpp).
- Just a reminder, InDesign has a list of UI colors that you can query to draw in UI, separate and different from the swatches you use in the document.
- Also a reminder, GetFrame() method in IControlView returns position of the widget in parent coordinates, you may want to use MoveTo(0,0) to transform coordinates.




Drawing framework : Dynamics

- **Offscreen drawing**
 - The purpose
 - Reduce flicker and improve performance
 - Layout offscreen
 - Background/foreground buffer and selection
 - Offscreen data
 - kOffscreenViewPortBoss
 - IPlatformOffscreen: the bitmap
 - Snapshots: memory based viewport
 - SnapshotUtilsEx

2005 Adobe Systems Incorporated. All Rights Reserved. 38




- The purpose of offscreen drawing is to reduce flicker and improve performance. An offscreen is simply a bitmap. when InDesign attempts to draw to a document window, it is actually drawing to the offscreen buffer. Then it periodically transfers its offscreen buffer to the actual screen.
- InDesign maintains an offscreen representation of the layout window with everything except the current selection highlighting. Hence, when the selection changes, InDesign can very quickly redraw the screen from the offscreen buffer followed by drawing the new selection.
- The most important data structure of offscreen drawing is the kOffscreenViewPortBoss. Everything supposed to draw to screen is drawn to this viewport. The offscreen bitmap is wrapped in platform-independent IPlatformOffscreen on kOffscreenViewPortBoss.
- Snapshot is a record of a part of document view at a specific moment. It is conceptually the same as offscreen drawing, but it has its own bitmap that can be exported to various graphic file formats, such as JPEG, TIFF or GIF. InDesign's export to JPEG is implemented using snapshot mechanism.
- There are two versions of snapshot, SnapshotUtils is based on CS 1, SnapshotUtilsEx is an improved version in CS2, which we recommended to use in future implementation. See snapshot sample plug-in for usages.




Drawing framework : Draw event handler

- **Custom draw event handler**
 - The purpose
 - To participate in drawing process
 - The architecture
 - Drawing events are broadcasted at specific points in page item drawing process
 - Register draw events
 - Service provider
 - Call `IDrwEvtDispatcher::Register()` directly

2005 Adobe Systems Incorporated. All Rights Reserved. 39




- You can participate in page item drawing process by providing a custom draw event handler. Drawing events are messages that are broadcasted at specific points in the drawing process. See page item drawing process slide.
- Your event handler can interrupt the drawing process by catching these messages, do your own drawing and then return to the drawing process. Depending on when you interrupt, you could draw your own stuff before or after original drawing.
- You will need to register your event handler to make it work. There are two way to register, either defines the event handler as a service provider that provides `kDrawEventService`; or, you can create the event boss and call `IDrwEvtDispatcher::Register()` directly. `IDrwEvtDispatcher` is on the `gSession` object.



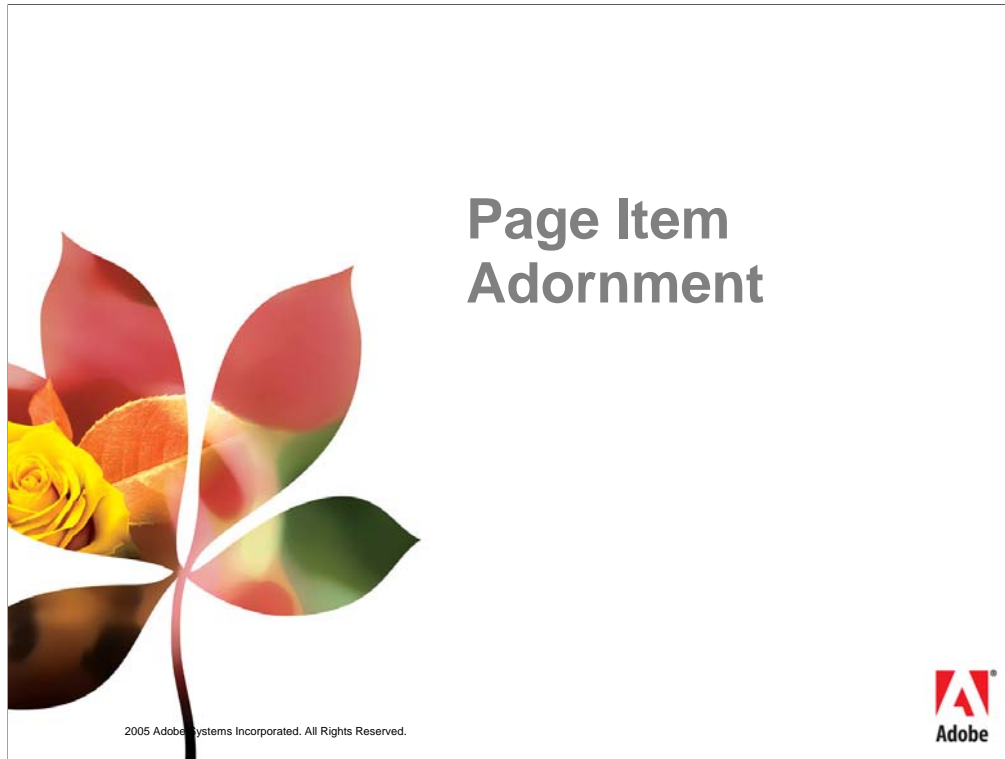
Drawing framework : Draw event handler

- **Custom draw event handler (continued)**
 - The implementation
 - Define drawing event handler boss class
 - Implement `IDrwEvtHandler::HandleEvent()`
 - Be sure to return `kFalse` if you want other drawing process resumes after you finish
 - Sample code
 - `PrintSelection`, basic draw event handler


2005 Adobe Systems Incorporated. All Rights Reserved. 40



- To implement a custom draw event handler, you will need:
 - Define draw event handler boss class in the resource. If you plans to register as service provider, remember to include the service provider, see example of “Basic Draw Event Handler” sample plug in, otherwise, don’t forget to hook event handler up directly, see “Print Selection” sample plug in
 - Then implement `IDrwEventHandler`. You will provide register and unregister method to identify the events you are in interested in. In the body of `HandleEvent()` method, make sure to return `kFalse` if you want drawing process to continue. For example, in `PrnSelDrawHandler::HandleEvent()`, return `kFalse` for those items that are selected to print.




- Now let's look at page item adornment




Page item adornment: Overview

- **The data model**
- **Custom page item adornment**

2005 Adobe Systems Incorporated. All Rights Reserved. 42




- Page item adornments customize the appearance of a page item, giving a plugin the chance to add additional drawing when the page item is rendered. We will talk about data model of the page item adornment, and how to custom a page item adornment.



Page item adornment: Data model

- **The adornments**
 - Independent boss objects
 - Implements IAdornmentShape or IAdornmentHandleShape
- **IPageItemAdornmentList**
 - Aggregates on kPageItemBoss
 - Referencing to page item adornments

2005 Adobe Systems Incorporated. All Rights Reserved. 43



- Similar to graphic attributes, page item adornments are stored as independent boss objects. A page item adornment implements either IAdornmentShape or IAdornmentHandleShape. All adornments a page item has are stored in IPageItemAdornmentList aggregated on kPageItemBoss.



Page item adornment: Custom page item adornment

- **How:**

- Provide adornment boss implementation and add to IPageItemAdornmentList

- **The drawing process**


- Adornment drawing can be called before or after drawing a page item shape or other events defined in the adornments.
- IAdornmentShape for shape, IAdornmentHandleShape for selection handle.

2005 Adobe Systems Incorporated. All Rights Reserved.

44




- To customize a page item using adornment, you have to define an adornment boss implementing IAdornmentShape or IAdornmentHandleShape and add the boss object to IPageItemAdornmentList of that page item.
- Page item adornment is drawn during page item drawing process. Adornment drawing methods can be called before or after drawing a page item shape or other events defined in the adornments. (But do not need to create custom draw event handler to draw adornments)
- Note IAdornmentShape::Draw() is called when drawing page item shape, IAdornmentHandleShape::Draw() is called when drawing selection handle.




Summary

- **CS1 graphics training review**
- **Graphic page items**
- **Color management**
- **Advanced graphic attributes**
- **Drawing**
- **Page item adornment**

2005 Adobe Systems Incorporated. All Rights Reserved. 45




- In summary, we have reviewed CS1 graphic training materials, talked about graphic page items' data model and exporting to graphic file format.
- We also briefly touched on color management basics.
- We also talked more in detail about several advanced graphic attributes, including rendering attributes, stroke effects and transparency.
- We covered in detail about drawing mechanism of InDesign, how layout, page item, UI and offscreen are drawn, how to customize draw event handler, how to decorate page item by adding page item adornments.



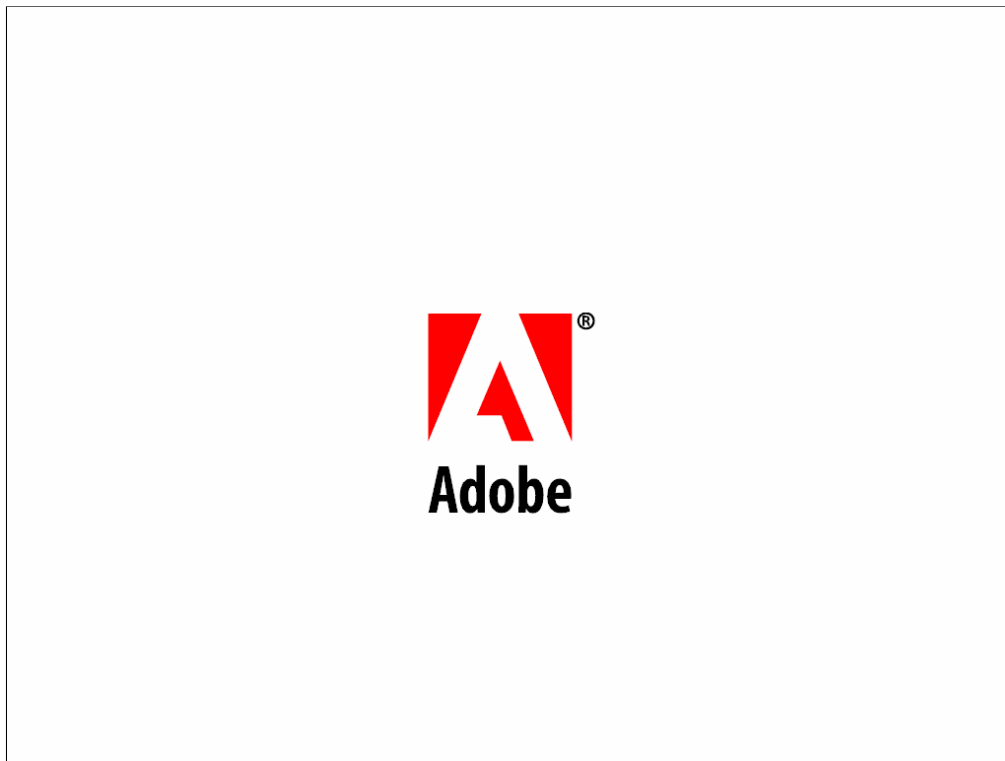
For more information...

- **The SDK**
 - Browsable reference documentation
 - ***InDesign CS2 Programming Guide***
 - Solution document: Working with graphics
 - Sample plugins and snippets

2005 Adobe Systems Incorporated. All Rights Reserved. 46



- For more information about graphics topic, I would recommend you to take a closer look at browsable reference documentation, InDesign CS2 programming guide: Graphic Fundamentals and Layout Fundamentals. The solution document “Working with graphics” has a lot of examples and use cases that you may want to look at before setting out to write your code. We also have plenty of sample plugins and snippet codes that illustrates that usage of graphic related interfaces and commands.



Thank you for your time for taking this training session. Happy developing your CS2 plugins.

