

An abstract graphic consisting of several overlapping, curved, translucent green ribbons that create a sense of depth and movement. The ribbons are rendered with a gradient from light to dark green and have a fine grid pattern on their surface.

3D / Forms Interaction



Adobe® Acrobat® 3D



3D / Forms Interaction

Created: 6/18/2006 | 1.0

Introduction

It can be very useful for a 3D Annotation to interact with a Dynamic Adobe Form in another document.

This can be used for:

- Creating more user friendly order forms where customers can visually select desired components instead of remembering names or part numbers.
- Providing extra repair or specification information associated with a particular 3D rendering of a product or product component.
- Increasing clarity in RFP bids on specific parts required in bidding process.

This tutorial demonstrates how to set up this type of interaction and gives a simple, hands-on example where the user can select parts of a 3D model in a PDF document and add them to a dynamic order form which is attached to the 3D PDF.

Key Concepts and Steps Covered

- 1.) Communicating between PDFs
- 2.) Prepare the 3D model for part selection
- 3.) Setup the part names in the 3D script
- 4.) Completing the Example

Terminology in this document

Two Adobe PDFs will be referred to in this document. The first will be referred to as the 3D PDF or the PDF containing the 3D Annotation. That document contains the 3D model and all the javascript required for the 3D interaction. The other PDF will be referred to as the Dynamic Adobe Form or the Dynamic Order Form and contains the ordering form and the logic associated with it.

3D / Forms Interaction

Created: 6/18/2006 | 1.0

Pre-Conditions

In order to complete the steps described in this document, you must have a basic knowledge of how to use **Adobe Acrobat** and **Adobe LiveCycle Designer**. You also must have access to a 3D model. This model can be imported from a CAD program, captured from any OpenGL 3D rendering application, or created using a 3D authoring tool. A basic knowledge of Javascript will help when implementing some parts of the functionality.

References

These are a few resources to help you understand how to utilize Javascript within Acrobat 3D and Adobe LiveCycle Forms Designer.

A. [Acrobat JavaScript Scripting Guide](#) (PDF: 2.52M)

This guide provides you with an overview of how you can use Acrobat JavaScript to develop and enhance standard workflows, and contains detailed descriptions of what the Acrobat JavaScript capabilities are and how to access them.

B. [Acrobat JavaScript Scripting Reference](#) (PDF: 7.12M)

This document provides detailed descriptions of all objects, properties and methods within the Acrobat extension to JavaScript, and contains code examples.

C. [Acrobat 3D JavaScript Reference](#) (PDF: 966k)

This document provides a detailed description of the new JavaScript objects, properties and methods to manipulate 3D content.

D. [Converting Acrobat JavaScript for Use in LiveCycle® Designer Forms](#) (PDF: 549k)

This document explains how to convert JavaScript contained in Adobe® Acrobat® Professional and Acrobat Standard forms for use in Adobe LiveCycle® Designer forms

Source Page: http://partners.adobe.com/public/developer/acrobat/sdk/index_doc.html#js

1. Communicating between PDFs

How it's Done

There are several ways for one PDF to communicate with another but one of the simplest and most reliable ways is to have one document directly call javascript functions embedded in the other document.

The entire form and subform structure of a Dynamic PDF is accessible through the *xfa* object of each Document. So calling the javascript functions of a Dynamic PDF from another document can be done like this:

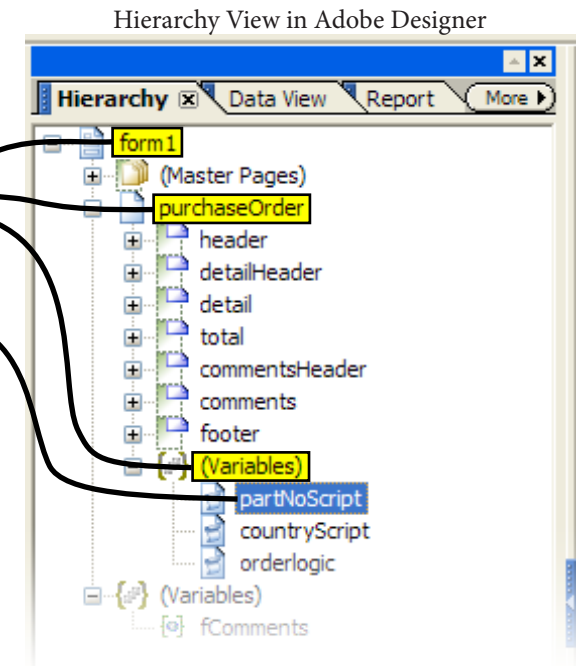
Example:

```
app.activeDocs[1].xfa.form.form1.purchaseOrder.variables.partNoScript.  
FunctionName(parameters);
```

General:

```
app.activeDocs[1].xfa.form.<form hierarchy>.<function name>(parameters);
```

Everything after “*xfa.form*” is directly from the “Hierarchy” view in Adobe LiveCycle Designer. As the diagram shows, just type out the hierarchy of forms and sub-forms to access the script object (Note the lowercase “variables”). The first part “*app.activeDocs[1]*” gets the document object for the Dynamic PDF (assuming it is already open and was opened after the 3D PDF).



Dealing with opening a specific PDF Document is a tricky thing. There are several ways to go about it.

1. Assume the document is already open and therefore part of the active docs

This is not a very good idea for an end-user document, you would need to include instructions telling the user to open an additional document increasing the probability that the interaction won't work.

2. Assume the needed PDF is in the same folder as the currently open document

This is also not good practice because the user could save the needed document in a different folder, or not have it at all.

Continues ...

3D / Forms Interaction

Created: 6/18/2006 | 1.0

... Continued

1. Communicating between PDFs

3. Include the needed document as an attachment to the PDF.

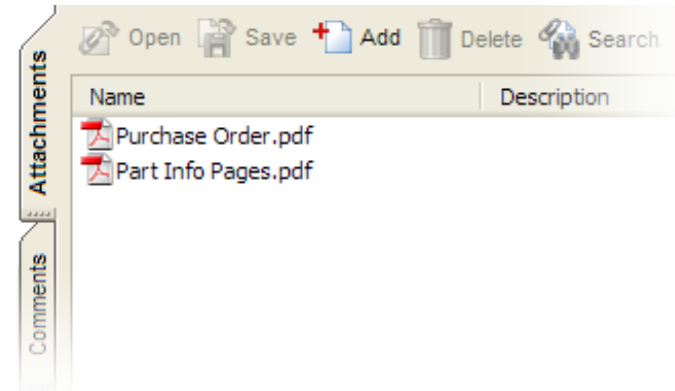
This is your best bet, it allows the 3D PDF and the Dynamic order form to be packaged in one file¹.

A PDF file included as an attachment in another PDF can be opened and displayed using javascript. These javascript functions do just that. If you add them to any script in the Advanced -> Javascript -> Document Javascripts collection, you can just call `OpenFile()`² to open an attached PDF and return the Document Object.

```
function OpenFile(attachmentname, documentname)
{
    //See if the document is already open
    var oDoc = FindOpenDoc(documentname);
    if (oDoc == null)
    {
        this.exportDataObject({cName: attachmentname, nLaunch: 2});
        oDoc = FindOpenDoc(documentname);
        app.execMenuItem("TileVertical");
    }

    return oDoc;
}

function FindOpenDoc(documentname)
{
    var oDoc = null;
    var docs = app.activeDocs;
    for(i in docs)
    {
        if(docs[i].documentFileName == documentname)
        {
            oDoc = docs[i];
            break;
        }
    }
    return oDoc;
}
```



¹ The attachment is extracted to a temporary directory and opened from there. Any changes made (and saved) by a user will be lost the next time the attachment is opened

² The argument *attachmentname* is usually "Untitled Object X" where X is the position the attachment appears in the Attachments panel starting with 1. The argument *documentname* is the file name of the attached PDF i.e. "Purchase Order.pdf".

Continues ...

1. Communicating between PDFs

... Continued

Now to illustrate this communication between documents, we will create an example that allows the user to select a portion of a 3D model (i.e. a part of a disc brake assembly) and add it to an order form in a Dynamic PDF.

1.1. Open the Sample

We will start with a slightly modified³ version of one of the examples that comes with **Adobe LiveCycle Designer**. Save a copy of the “Purchase Order.pdf” that came attached to this tutorial (View -> Navigation Tabs -> Attachments). Launch **Acrobat LiveCycle Forms Designer** and open up the PDF you just saved a copy of.

1.2. Create the “AddItem” function

Add a Script Object to the “Purchase Order” subform and name it “orderlogic”. Paste this code in the script object to create an AddItem function.

```
function AddItem(PartNo)
{
    if(PartNo == `` || PartNo == null)
        return;

    _detail.addInstance();
    var detailForm = purchaseOrder.detail.all.item(_detail.count - 1);
    detailForm.txtPartNum.rawValue = PartNo;

    //To recalculate the order totals
    xfa.form.calculate(1);
}
```

³ The attached PDF is a modified version of the Designer example usually located at:

C:\Program Files\Adobe\Acrobat 7.0\Designer 7.0\EN\Samples\Purchase Order\Dynamic Interactive\Forms\

Continues ...

... Continued

1.3. Set the Disclosed Property

You have to add the following code to the *docReady* event of the main form object (form1 in this case) to allow other documents to call the javascript functions in this one.

```
event.target.disclosed = true;
```

Then save this Document as “Purchase Order.pdf”

1.4. ... On to the 3D PDF

Either create a new PDF or open an existing one in **Adobe Acrobat 3D**. Create a button with the title “Add To Order” and create a “Run a Javascript” action for the button with the following code.

```
var context3D = getAnnots3D(0)[0].context3D;
if(context3D.SelectedLayerName != '')
{
    var oDoc = OpenFile("Untitled Object 1", "Purchase Order.pdf");

    if(oDoc != null)
        oDoc.xfa.form.form1.purchaseOrder.variables.orderlogic.AddItem(context3D.SelectedLayerName);
}
```

Click the menu: Advanced -> Javascript -> Document Javascripts. Type in a script name, “DocOpen” for instance and click “Add”. Replace the contents of the script editor with the two functions(“OpenFile” and “FindOpenDoc”) from the end of the “How It’s Done” section.

3D / Forms Interaction

Created: 6/18/2006 | 1.0

2. Prepare the 3D Model for Part Selection

To make setting up the part selection functionality easier, the hierarchy of the model should be arranged so that the 3D objects are grouped by part.

The first step is to determine which parts of your model you want to let the user add to the order.

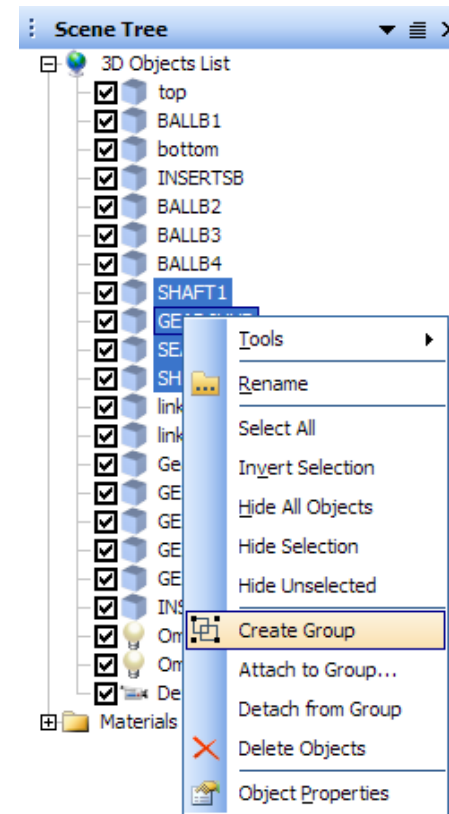
Open the model in Adobe 3D Toolkit and use the grouping features to re-arrange the parts of the model in the hierarchy.

1. For each part that you want the user to be able to select independently, select all the 3D objects that make up the part.

2. Right click on one of the selected 3D objects and select Create Group.

3. Right click again and select “Rename” to give the new group an appropriate name.

4. Repeat this process for each orderable part and then save the model under a new name.



3. Setup the part names in the 3D script

Now that the model is arranged by part, making the parts selectable is fairly easy. Save a copy of the “PartSelection.js.txt” script that came attached to this tutorial (View -> Navigation Tabs -> Attachments). Make sure to save it as “PartsSelection.js”, removing the “.txt”, then open it up for editing. Only the section titled “Model Specific Section” needs to be changed.

This script allows the 3D objects within the scene to be grouped into selectable layers. Each layer can then be controlled via javascript calls to a LayerManager class. In this script the selecting and highlighting of layers is done automatically. When a layer is clicked, the value in the Layer’s *name* property is copied to the public *SelectedLayerName* variable so it can be accessed from within document scripts. There is also a method called *SetLayer-ChangedHandler* that you can use to provide a method that should be called when a new Layer is selected.

3.1. Write the code to setup the Layers

For each part of the model you want the user to be able to select, add a line of javascript code to the “Model Specific Section” that adds the appropriate 3D objects to a layer. Multiple 3D objects can be added to each layer.

```
layerManager.addPartToLayerByName("Casing", 0);  
layerManager.getLayer(0).name = "CASE67";  
  
layerManager.addPartToLayerByName("Control Board", 1);  
layerManager.getLayer(1).name = "CB-174";  
  
layerManager.addPartToLayerByName("Shaft1", 2);  
layerManager.addPartToLayerByName("Shaft2", 2);  
layerManager.getLayer(2).name = "TS-SHAFTS";  
...
```

The *addPartToLayerByName(Name, LayerIndex)* function call adds the specified 3D Object to the specified layer. The 3D Object can be the name of a Mesh or a Group that you created on the previous page. The *getLayer(0).name = 'CASE67'* line specifies the text string that will be copied to the *SelectedLayerName* variable. The layer name is rather arbitrary, but in the case of an order form, it is helpful to name the layers with catalog part numbers.

4. Completing the Example

4.1 Assembling The Example

1. Launch **Adobe Acrobat 3D** and open the PDF you created or used in Step 1.4.
2. Create a new 3D Annotation on the first page of the document using the “3D Tool”.
3. Click “Browse” next to the “3D Model” box and point it to the 3D model you modified in Step 2.
4. Click “Browse” next to the “Default Script” box and point it to the script you edited in Step 3.1.¹
5. Click the menu: View -> Navigation Tabs -> Attachments
6. In the Attachments Tab, click “Add” and select the Purchase Order.pdf that you saved in step 1.3.
7. Save the 3D PDF Document

4.2 Using the Example

1. Open the PDF you just saved in step 4.7.
2. Change to the Hand Tool so you can interact with the 3D scene.

Note: You should now see the different parts of the 3D scene highlight as you move the mouse over them.

3. Click one of Parts in the 3D scene

Note: The part you clicked should now remain highlighted

4. Click the “Add To Order” button that you created in Step 1.4.

The Dynamic Order Form should open up and the selected part number should be added to the list on the Dynamic Order Form PDF.

¹ You can reattach the script at a later time if you make changes. Double click on the 3D Annotation (while in editing mode) and click the “Edit Content” button.

4. Completing the Example

4.3 Extending the Example

To make this example a bit more realistic, part numbers, descriptions, and prices can be added to the “partNoScript” in the Dynamic PDF using Adobe Designer. The description and price fields will then automatically be filled in when a part is added to the order.

The Part number, description, and prices arrays look like this:

```
var partNo = new Array(" ",
    /* Add Part Numbers here */
    "580463116",
    "25906311C",
    ...

var partDesc = new Array(null,
    /* Add Part Descriptions here */
    "Electric Fuel Pump",
    "Air Flow Meter",
    ...

var partPrice = new Array(null,
    /* Add Part Prices here */
    149.95,
    145.95,
    ...
```

Just add you own values underneath each `/* Add ... here */` comment separated by commas and then save and re-attach the Dynamic Order Form to the 3D PDF.