

## Animation with PDF3D XmlServer Tutorial

### 1. Introduction

This tutorial will give a brief introduction to creating animations through the workings of the PDF3D XmlServer, a piece of application software enabling the animation and manipulation of 3D objects within standard PDF files. PDF3D XmlServer is also available as a package within PDF3D-SDK Development Tools, which also includes PDF3DReportGen.

Throughout the tutorial, we will use a standard word processing package (such as Microsoft Notepad or your favourite editor) to create and edit .xml files, which are then “built” by the PDF3D XmlServer and PDF files are returned as output with a pre-specified file name. These output PDFs can then be subjected to further interactive manipulations using the free Adobe reader or Adobe Acrobat Pro Extended.

Our test object will be a pre-defined 3D model of an electric motor in 3DS (3D Studio Max) format, whose components will, in turn, be translated, rotated, scaled and be made transparent by our programming commands. Working will be explained in detail throughout.

### 2. Preliminaries

Firstly, open the file “motor\_animation1.xml” using a standard word processing package. Since this is only a brief tutorial, we will not concern ourselves with the background code in the first few lines<sup>1</sup>, and instead start by looking briefly at the paragraph beginning “<Assembly>”. Our predefined motor (to be animated) is the file “motor.3DS”, which is imported with the lines:

```
<InputFileName value="motor.3DS" />  
<NodeName value="MOTOR" />
```

The complete motor, as the union of its components, is given the name “MOTOR”. Grouping components together in this way becomes very useful, since often a simple object can have quite complicated graphics; in order for the object to perform a basic movement, all its components have to be co-ordinated, so grouping them together can save several lines of code.

Default parameters such as background colour, foreground colour and opacity can also be specified:

```
<DiffuseColor r="0" g="128" b="128" a="255" />  
<AmbientColor r="0" g="128" b="128" a="255" />  
<SpecularColor r="250" g="50" b="50" a="255" />  
<Opacity value="1.0" />
```

---

<sup>1</sup> If the reader is interested, the purpose of these lines is outlined in the file “input-specifications.xml”.

Here “r” stands for the amount of red, “g” for green, “b” for blue and “a” for the overall strength of colour, and values taken are integers between 0 and 255. In our file however, we will not worry about these and their purpose is removed with the brackets “<!--“ and “-->”. The principle reason is if we set these values here, colours will be ignored from the file.

We set the initial viewing angle by specifying a rotation about a chosen axis (given by a unit vector) by a chosen angle (with positive orientation), and the camera “zoom” with the lines:

```
<Scale type="manual" scale="1.5" scalex="1.5" scaley="1.5"
scalez="1.5"/>
<Orientation vectorx="0.497" vectory="0.497" vectorz="0.711" angle="-
1.57" />
```

Note: vector should be specified as unit length, else you may experience unexpected sheer transformation.

### **3. Translation**

Now try building the file “motor\_animation1.xml”. To do this, one can use the DOS prompt and one the correct directory is located, execute the command “PDF3DXmlServer.exe motor\_animation1.xml”. The resultant PDF file “motor\_animation1.pdf” will then be built as specified by the third line of code:

```
<OutputFileName value="motor_animation1.pdf" />
```

Now open the PDF. If the XmlServer has worked correctly, you should be looking at a motor from an angle, with its front face being gradually removed.

Our first lines of code to specify movement are:

```
<NodeMovement nodeName="PolyFace75_0" localRepeatMode="none">
  <KeyFrame>
    <Position positionx="0" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="0" vectorz="1" angle="0"
      centerx="0" centery="0" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="5.0"/>
  </KeyFrame>
  <KeyFrame>
    <Position positionx="10" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="1" vectorz="0" angle="0"
      centerx="0" centery="0" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="0.0"/>
  </KeyFrame>
</NodeMovement>
```

The line

```
<NodeMovement nodeName="PolyFace75_0" localRepeatMode="none">
```

specifies what to do with the component “PolyFace75\_0”, which corresponds to the front circular face of the motor, as can be seen by highlighting this component in the model tree in

the PDF file. If you need to find the names of any object, just select them with your left mouse button while the “Model Tree” is open to reveal their names. Since the other components present in the motor pass unmentioned in the code, nothing happens to them, and they remain in their default positions. We will turn to the significance of the “localRepeatMode” later.

The movement is characterised by specifying “*key frames*”, which are positions which the face must be at after a specified time delay. In our instance, there are two key frames, one where the face is in its initial position, attached to the rest of the motor, and one where it is completely removed. In the code between each “<KeyFrame>” and “</KeyFrame>” we specify the face’s global position, rotation (about a chosen axis) and scale.

In this file, we have constructed a simple translation in the positive x-direction by specifying one key frame in the initial position, and one with the motor’s face centred on (10,0,0) with the lines

```
<Position positionx="0" positiony="0" positionz="0"/>
```

and

```
<Position positionx="10" positiony="0" positionz="0"/>
```

respectively<sup>2</sup>, with a “delay” of 5 seconds to get from one key frame to the next given by the line “<Delay value="5.0"/>”. At the end, the movement is looped back around.

#### 4. Multiple Movements

Of course, consecutive movements are possible. Open the file “motor\_animation2.xml” and build it. Here there are two movements made by the motor’s face: first it detaches itself and moves in the positive x-direction as before, but then it moves perpendicularly in the positive y-direction. Hence three key frames are required to specify its position after each translation, as can be seen from the code:

```
<NodeMovement nodeName="PolyFace75_0" localRepeatMode="none">
  <KeyFrame>
    <Position positionx="0" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="0" vectorz="1" angle="0"
      centerx="0" centery="0" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="5.0"/>
  </KeyFrame>
  <KeyFrame>
    <Position positionx="10" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="1" vectorz="0" angle="0"
      centerx="0" centery="0" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="5.0"/>
  </KeyFrame>
  <KeyFrame>
    <Position positionx="10" positiony="10" positionz="0"/>
    <Rotation vectorx="0" vectory="1" vectorz="0" angle="0"
      centerx="0" centery="0" centerz="0"/>
  </KeyFrame>
</NodeMovement>
```

---

<sup>2</sup> The movement required, given the starting and finishing positions, is always given by a linear transformation calculated by the program.

```

    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="5.0"/>
</KeyFrame>
<KeyFrame>
    <Position positionx="10" positiony="10" positionz="0"/>
    <Rotation vectorx="0" vectory="1" vectorz="0" angle="0"
        centerx="0" centery="0" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="0.0"/>
</KeyFrame>
</NodeMovement>

```

Five second delays are specified between each key frame, as before.

## 5. Multiple Components

Now open the file “motor\_animation3.xml” and build it. In turn, the motor face (“PolyFace75\_0”), the motor handle (“PolyFaceM0\_0”) and the motor shaft (“Cilindro01\_0” and “Cilindro01\_1” combined) move in perpendicular directions, so (after a moment’s thought) this requires four key frames. However, the difference to the previous situation is that we need multiple sections of the form

```

<NodeMovement nodeName="..." localRepeatMode="none">
    ⋮
</NodeMovement>

```

for each component. Note that we need not have included lines such as

```

<KeyFrame>
    <Position positionx="10" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="1" vectorz="0" angle="0"
        centerx="0" centery="0" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="5.0"/>
</KeyFrame>
<KeyFrame>
    <Position positionx="10" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="1" vectorz="0" angle="0"
        centerx="0" centery="0" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="5.0"/>
</KeyFrame>
<KeyFrame>
    <Position positionx="10" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="1" vectorz="0" angle="0"
        centerx="0" centery="0" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="0.0"/>
</KeyFrame>

```

where an object is resting between key frames; each component can be given its own key frames entirely independently, hence the above code could be cleaned up by having just two

key frames for the node with a delay of 10 seconds. The code is only written in this way to make clear the various stages of movement.

## 6. Pingpong and Smooth Return

Currently, after a given component finishes its prescribed movement, it returns instantaneously to the first key frame specified, and loops back round. This has to do with the command “localRepeatMode=“none”” within each node movement, and the overriding command “repeatModeByDefault=“none”” found below the assembly code.

Both “localRepeatMode” and “repeatModeByDefault” can take one of three values: “none”, “pingpong” or “smooth”. localRepeatMode applies to the individual node movement the code is contained in, and repeatModeByDefault overrules whatever was fed into the localRepeatMode, and applies to every moving component<sup>3</sup>.

In our current file, both are set to “none”, which means that the moving components snap back instantaneously to the starting position. To illustrate the effect of “pingpong”, open “motor\_animation4.xml” and build it. Here “pingpong” is applied to repeatModeByDefault. At the end of the cycle of movement, the motion is inverted and the objects return to their initial positions in the reverse order to which they began moving. The objects also move at the same speeds as previously, so the time taken to return is the same as the time taken to reach the end key frame. Note that the individual delay times for each node at their final key frames are ignored.

For an example of “smooth”, open “motor\_animation5.xml”. Here the objects return smoothly to their initial key frames in the times specified by the delay times at the final key frames<sup>4</sup>.

## 7. Rotation

We will now demonstrate how objects can be rotated. Open “motor\_animation6.xml” and build it. The code below governs a 360° rotation of the motor’s handle:

```
<NodeMovement nodeName="PolyFaceM0_0" localRepeatMode="smooth">
  <KeyFrame>
    <Position positionx="0" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="0" vectorz="1" angle="0"
      centerx="0.1" centery="1" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="2.0"/>
  </KeyFrame>
  <KeyFrame>
    <Position positionx="0" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="0" vectorz="1" angle="2.09"
      centerx="0.1" centery="1" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="2.0"/>
  </KeyFrame>
  <KeyFrame>
    <Position positionx="0" positiony="0" positionz="0"/>
```

---

<sup>3</sup> In order to cancel its overriding effect, the command must be removed completely.

<sup>4</sup> This is done via a default linear transformation, independent of the movements carried out to reach the final key frame.

```

    <Rotation vectorx="0" vectory="0" vectorz="1" angle="4.19"
      centerx="0.1" centery="1" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="2.0"/>
  </KeyFrame>
  <KeyFrame>
    <Position positionx="0" positiony="0" positionz="0"/>
    <Rotation vectorx="0" vectory="0" vectorz="1" angle="6.28"
      centerx="0.1" centery="1" centerz="0"/>
    <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0"/>
    <Opacity value="1.0"/>
    <Delay value="0.01"/>
  </KeyFrame>
</NodeMovement>

```

With respect to a default position and orientation, the code

```

<Rotation vectorx="0" vectory="0" vectorz="1" angle="2.09"
  centerx="0.1" centery="1" centerz="0"/>

```

specifies the position of the node given by a positively oriented rotation of 2.09 radians (approximately 120°) about the axis with unit direction (0,0,1), passing through the point (0.1,1,0). The point (0.1,1,0) happens to correspond with the centre of the motor’s handle, so its global position remains unchanged.

Hence, taking the delay values into account, our programming specifies a rotation from 0 to 2.09 radians in 2 seconds, then from 2.09 radians to 4.19 radians in 2 seconds, then from 4.19 to 6.28 radians in 2 seconds<sup>5</sup>. Since 6.28 radians is not quite 360°, the following delay value of 0.01 seconds and the setting “localRepeatMode=“smooth”” indicate a short, minute movement back to the starting position.

## 8. Scaling

Open the file “motor\_animation7.xml” and build it. This is a demonstration for how objects can be scaled, with a slight variation to before. Instead of prescribing movement to individual components, we have applied our transformations to the entire model (“MOTOR”) with the line:

```

<NodeMovement nodeName="MOTOR" localRepeatMode="smooth">

```

Look at the model tree in the output PDF. Here, “MOTOR” represents the entire image, with each of its components as subdirectories, so prescribing actions for “MOTOR” gives the desired effect. More generally, any combination of components can be grouped together to facilitate simultaneous movement.

The code within each key frame specifying scaling is of the form

```

<Scale scale="..." scalex="..." scaley="..." scalez="..." />

```

“scale” represents the overall scale factor, multiplying each co-ordinate of the node by the value in quotation marks and “scalex”, “scaley” and “scalez” represent the individual scalings about the x, y and z-axis respectively. Values of 1.0 represent no scaling. As before, the

---

<sup>5</sup> Angles must have magnitude less than  $\pi$  radians, hence the need for three separate rotations.

scaling is not immediate, and the model transforms smoothly from its initial position to the scaled position in the specified delay time.

After each scaling (a scaling is executed in the x, y, then z-direction respectively, followed by an overall scaling), then model returns to its initial position by including a key frame with the default scaling of 1.0 in all directions.

## **9. Opacity**

Finally we will show how objects can be faded in and out of the image by changing their opacity. Open and build the file “motor\_animation8.xml”. Within each node movement and key frame, the opacity of the relevant node is governed by the line:

```
<Opacity value="..." />
```

Here the numerical value within the quotation marks can be any number between 0 and 1, with 0 being invisible (i.e. totally transparent) and 1 being completely opaque. Just as with previous operations, the program transforms the object in question smoothly from one key frame to the next, so in our file, the code

```
<KeyFrame>
  <Position positionx="0" positiony="0" positionz="0" />
  <Rotation vectorx="0" vectory="0" vectorz="1" angle="0"
    centerx="0" centery="0" centerz="0" />
  <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0" />
  <Opacity value="1.0" />
  <Delay value="5.0" />
</KeyFrame>
<KeyFrame>
  <Position positionx="0" positiony="0" positionz="0" />
  <Rotation vectorx="0" vectory="1" vectorz="0" angle="0"
    centerx="0" centery="0" centerz="0" />
  <Scale scale="1.00" scalex="1.0" scaley="1.0" scalez="1.0" />
  <Opacity value="0.0" />
  <Delay value="5.0" />
</KeyFrame>
```

specifies a fading-out of the component in question (with no physical movement) in a duration of 5 seconds. “pingpong” repeat mode has been specified globally, so once the prescribed fadings have been executed, the program inverts them.

As a demonstration for all that has been covered in this tutorial, the file “motor\_animation9.xml” incorporates translation, rotation, scaling and opacity variation to each of the components previously transformed. Build it and see for yourself.



## **10. Final Result**

The resulting animation 3D PDF file produced by this tutorial may be viewed at:  
[http://www.pdf3d.com/download\\_pdf.php?file=motor\\_animation9.pdf](http://www.pdf3d.com/download_pdf.php?file=motor_animation9.pdf)