IM473688

Get More out of your Engineering Data by Visualization

Simon Nagel

Autodesk

Melanie Thilo
Autodesk

Przemek Sokolowski
Autodesk

Learning Objectives

- Discover the benefits and requirements of a good visualization
- Learn how to create a compelling story based on your Inventor data with VRED.
- Learn how to establish a workflow between Inventor and VRED—you can use visualization in your design process without preparation time
- Learn how to use a live link between Inventor and VRED for iLogic Configuration, Geometry Update and Animations

Description

A compelling visualization is a key element for a successful presentation. Whether for an internal design review or an external customer-facing presentation, good storytelling is essential when working with any kind of complex data. This class will show you how to get more of your Inventor engineering data with the powerful visualization of VRED software. You will experience how easily and quickly you can tell your story with your data with astonishing quality. Learn how to import, reference, manage, and update Inventor data natively in VRED, and choose if you want to create real-time scenes, photorealistic renderings, engineering lighting analysis, virtual reality experiences, portable EXE files, or cloud-based visual content. We are looking forward to showing you a seamless process between Inventor software and VRED, and demonstrating how to sync data, configurations, and transformation.

Speakers

Simon Nagel

Simon Nagel is an expert for Visualization for Realtime Rendering, High-quality-image generation and Virtual Reality. Simon is working in the Design industry for 15 years as 3D Artist, Consultant, UX Designer, Product Manager and Technical Sales Specialist. His main focus is the Automotive Industry.

Melanie Thilo

Melanie Thilo has worked for more than 10 years as a mechanical design engineer for different companies and industries. Since 2018 Melanie is a Technical Sales Specialist for PDMC (Inventor) at Autodesk.

Przemek Sokolowski

Przemek Sokolowski works for Autodesk as a Technical Sales Specialist in Poland since 2007. Before Autodesk he worked for several years in Man and Machine (Autodesk distributor in Poland, at that time). He graduated Technical University in Lodz, specializing in robot control system. Przemek is also well known in Poland for running the blog dedicated to Autodesk D&M products: "Po prostu Inventor”. Big fan of iLogic

Table of Contents

Introduction

Decision making in a digital environment

The benefits of digital prototyping are recognized more and more in the industry. By creating and testing new products in a CAx environment, issues can be detected and corrected earlier.

Traditionally the engineers in product development work with CAD software to author data and PDM/PLM systems to manage data. Sometimes communication workflows are implemented in these systems. And sometimes decision makers have no direct access this engineering data.

In design reviews, a new concept, a draft, or a new product is presented to a decision-making committee for approval. Frequently, the new product is presented to decision-makers by using Power Point presentations or a prototype is made especially for the event. Surprisingly rarely design or CAD data is used in design reviews.

The way decisions are made in a digital world is changing.

Challenges of digital decision making

Dimensions and proportions of a CAD-Model are very hard to judge. A car and a clockwork can have the same size on the monitor. Missing the ability to compare the dimensions with the own body.

The preparation of a physical Prototypes can be very expensive. Other presentation media like PowerPoint Presentations tend to be not very flexible, and preparation can be very time consuming.

To ensure the quality of a product issues should be detected as early as possible. This means all decision makers need accessible information.

Value of high-quality visualization

A good visualization gives confidence in digital decision Making.

Replicate Reality to save cost and time

Replication of reality can save cost and time. By using physical parameters a realistic digital replication of reality is created.

Quick Design Exploration and Presentation

Good Visualization gives easy access to quick design exploration and Presentation. Virtual reality enables stakeholders to understand complex digital data without software knowhow.

Visual Communication and Collaboration improve Workflow

Visual communication and collaboration improve the workflow. The user can choose the best media for presentation and provide all information of every model on every device.

Early Detection of Design Intent and Quality

collaborate across borders with stakeholders to detect problems as early as possible to ensure the quality of your product.

Solution

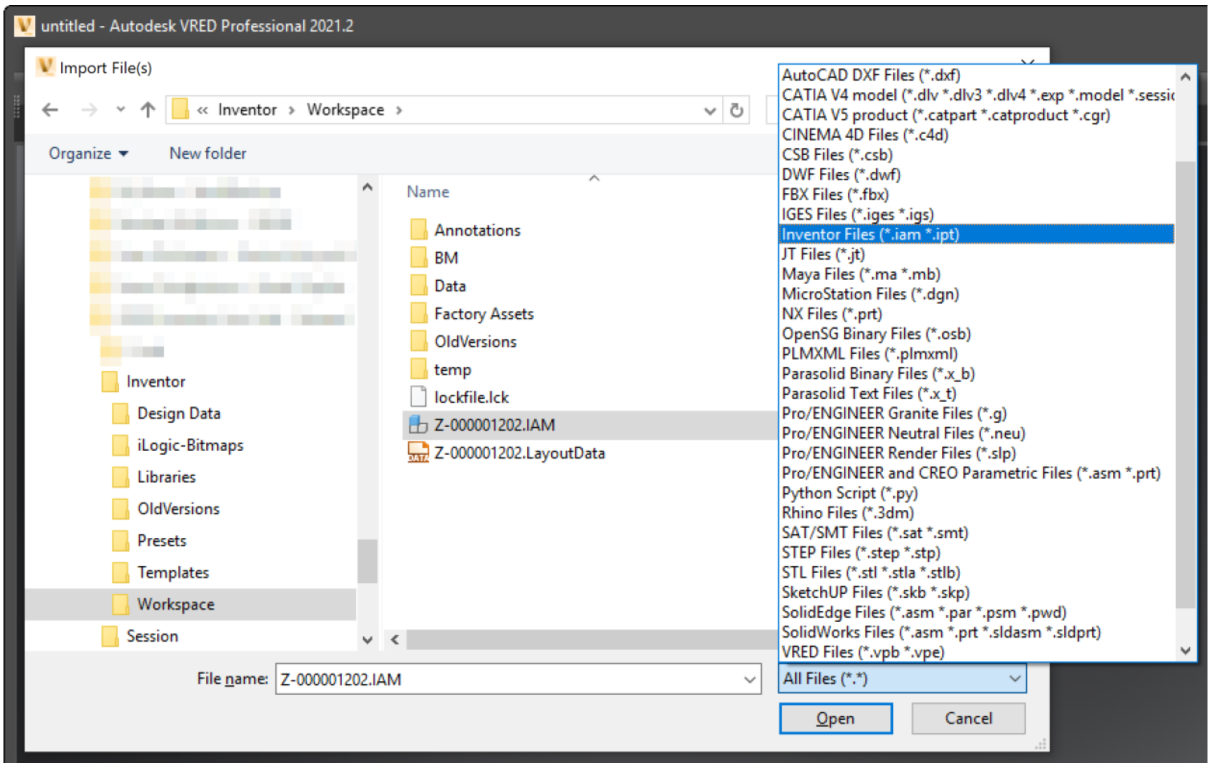
For an effective decision making process, an easy to set up and scalable workflow is needed.

Use Case 1 – Sales and Marketing Configurator

Create in four simple steps an easy to use configurator for sales and marketing by reusing engineering data to present your product to external stakeholders.

1. Import design data

VRED supports many native formats for direct import (see Figure 1). This saves time, because no translation is required to visualize design data.



untitled - Autodesk VRED Professional 2021.2

Import File(s)

« Inventor › Workspace ›

Organize ▾ New folder

- Inventor
  - Design Data
  - iLogic-Bitmaps
  - Libraries
  - OldVersions
  - Presets
  - Templates
  - Workspace
- Session

Name
- Annotations
- BM
- Data
- Factory Assets
- OldVersions
- temp
- lockfile.lck
- Z-000001202.IAM
- Z-000001202.LayoutData

AutoCAD DXF Files (*.dxf)
CATIA V4 model (*.dlv *.dlv3 *.dlv4 *.exp *.model *.sessic
CATIA V5 product (*.catpart *.catproduct *.cgr)
CINEMA 4D Files (*.c4d)
CSB Files (*.csb)
DWF Files (*.dwf)
FBX Files (*.fbx)
IGES Files (*.iges *.igs)
Inventor Files (*.iam *.ipt)
JT Files (*.jt)
Maya Files (*.ma *.mb)
MicroStation Files (*.dgn)
NX Files (*.prt)
OpenSG Binary Files (*.osb)
PLMXML Files (*.plmxml)
Parasolid Binary Files (*.x_b)
Parasolid Text Files (*.x_t)
Pro/ENGINEER Granite Files (*.g)
Pro/ENGINEER Neutral Files (*.neu)
Pro/ENGINEER Render Files (*.slp)
Pro/ENGINEER and CREO Parametric Files (*.asm *.prt)
Python Script (*.py)
Rhino Files (*.3dm)
SAT/SMT Files (*.sat *.smt)
STEP Files (*.step *.stp)
STL Files (*.stl *.stla *.stlb)
SketchUP Files (*.skb *.skp)
SolidEdge Files (*.asm *.par *.psm *.pwd)
SolidWorks Files (*.asm *.prt *.sldasm *.sldprt)
VRED Files (*.vpb *.vpe)

File name: Z-000001202.IAM

All Files (*.*)

Open Cancel

FIGURE 1

Learn more about the Import Files Dialog

After importing an *.iam-File in the Scenegraph of VRED the structure of the file can be accessed as known from the browser in Inventor.

Tessellation can also be adjusted afterwards to change Polycount, depending on the on the quality or performance requirements of a scene.

For Storytelling and Performance, the scene can be optimized by regrouping, attaching, detaching, merging surfaces and adjust normal.

Renderpeople from the Asset Library can be used to better assess dimensions and proportions. An example is shown in Figure 2.

Learn more about how working with the Assets and the Asset Manager

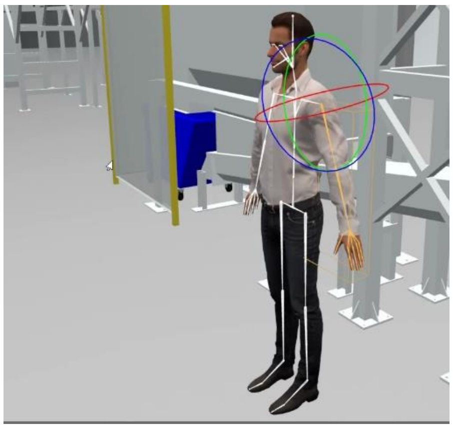


FIGURE 2

2. Adjust Materials and Look

The material assignment is used from the Inventor file. These materials can be easily adjusted in VRED for a more realistic look. A large online library of substance materials is available for photorealistic results. The textures are loaded and can be adjusted in real-time. Figure 3 shows an example of a substance material applied in the scene.

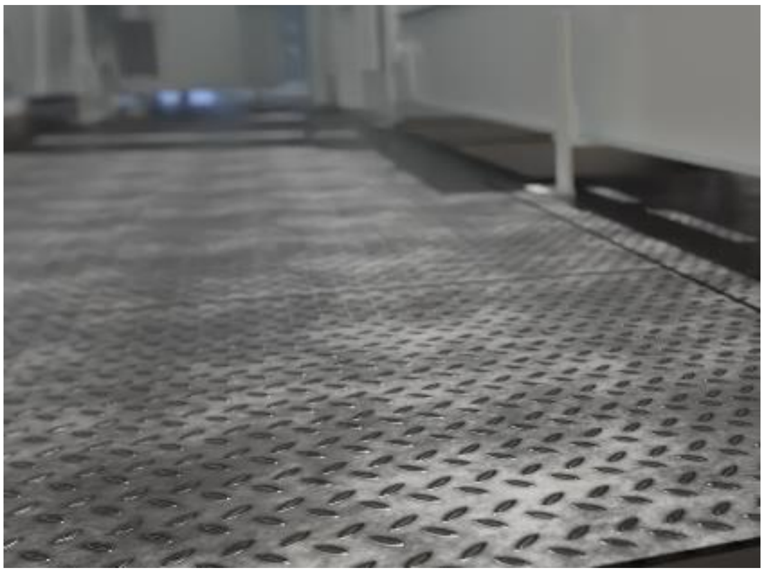


FIGURE 3

The environment material is defined for the for an attractive scene lighting. Here you can choose from different options like Skylight or HDR 360 Panorama.

Enable shadows for a more realistic look. Get real time shadow from sunlight or ambient occlusion. For a high visualization quality light, shadows and ambient occlusion can be baked. Compare Figure 4 and Figure 5 to see the difference.

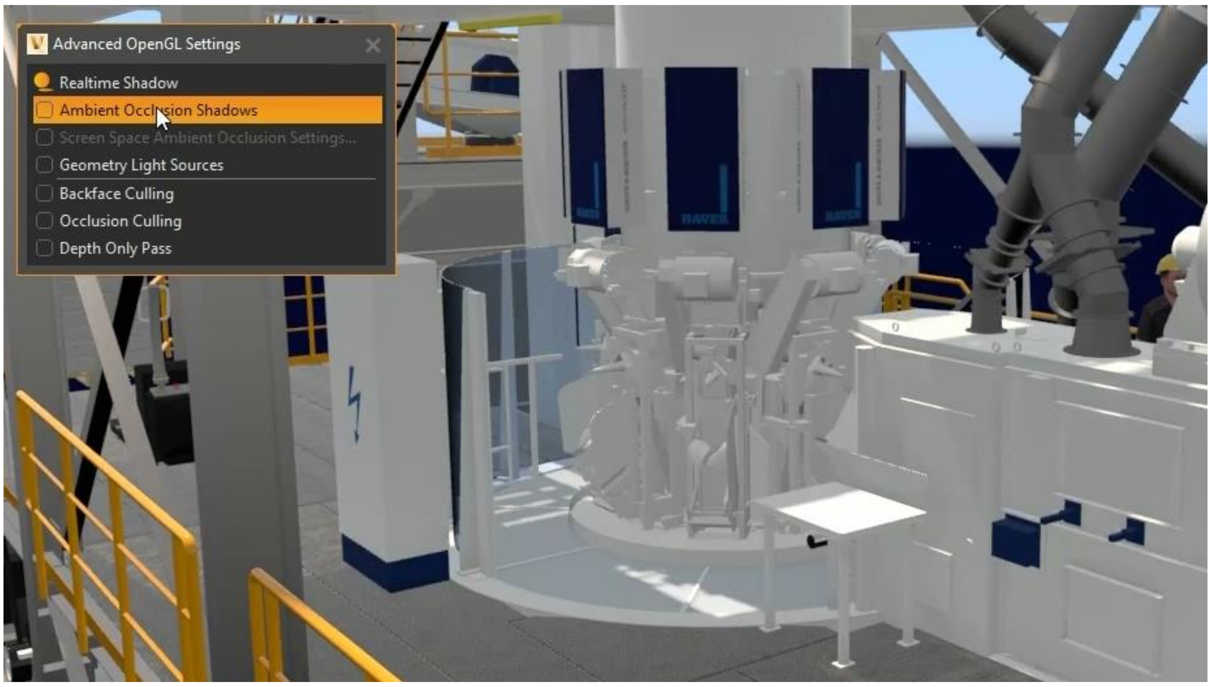


Figure 4

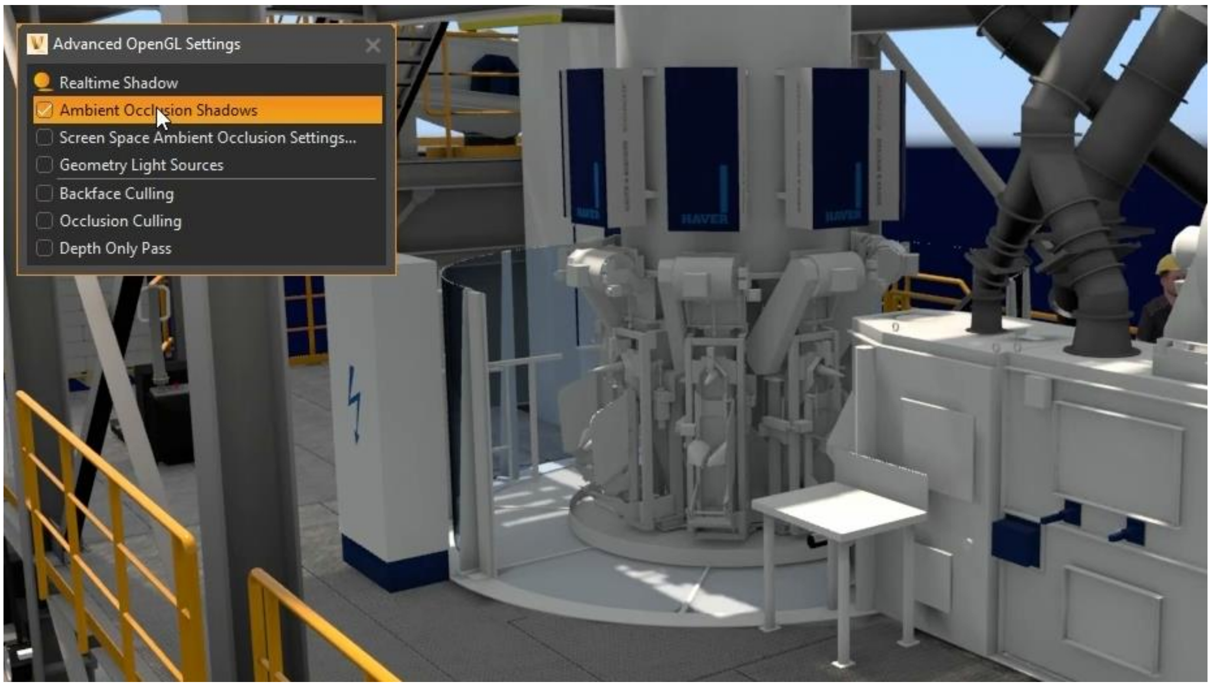


Figure 5

The entire process can be automized. As VRED is capable of using Python, a batch process can be used.

[Here is an example on how to use the post Python Command for Batch in VRED]

In that case you need scripts for:

a) import the data

```
filename = "D:/VRED-Live Link/Main Assy/Dust_Collector.iam"
vrFileIOService.importAtfFile(filename,vrNodeService.findNode("Root"))
```

b) calculate Shadows

```
selectNode("Root")
computeAmbientOcclusion(3, 0, 3000, 1, 0, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0)
```

c) assign materials from Asset manager.

```
assetpath =
"C:/Users/simon/Documents/Autodesk/Automotive/VRED/Assets/Materials/H&B"
applyMaterialAssetsByName(assetpath)
```

Photorealistic materials can be automatically assigned to the scene by using the Asset Manager.

3. Create a compelling storyboard

VRED includes different tools for storytelling.

- Viewpoints give fast access to key positions in the 3D scene.
  [Learn how to create Viewpoints]

- Sceneplates enrich a scene with meta-information.
  Learn more about Scene Plates
- Predefined Variants give quick access to different configurations.
  Learn how to create Variants
- Touch Sensors make a scene more interactive.
  Learn more about Touch Sensors

With these tools users have a wide range of options to create a compelling story. For example, a movie can be created that plays automatically, but can be paused or stopped at any time and a new storyline can be created.

This gives full flexibility to react individually to the audience.

4. Define Output

Once the scene has been prepared, the preferred output medium must be defined. VRED provides multiple different output options, Table 1 gives an overview of frequently used.

Multiple Output Options			
offline	**Still Frame Image** Realtime Quality or Photorealistic Quality	**Animated Movie** Realtime Quality or Photorealistic Quality Keyframe Animation can be created or imported	**360 Panoramas** Realtime Quality or Photorealistic Quality Embed in Webpages for wide distribution
online	**3D Interactive Realtime** Embed in Webpages for wide distribution	**Web Streaming** Streaming for Mobile Device Access and Realtime Navigation	**Virtual Reality** One Click access to Virtual Reality and Collaboration

TABLE 1

Additionally, a scene can be published as a VRED Go Executable. This file allows to explore the 3D data without any installation.

Learn more about VRED Go

Use Case 2 – Virtual Design Review

Of course, scenes can also be prepared for design reviews as described in *Use Case 1 – Sales and Marketing Configurator*. As the product is still in development, a closer connection to the CAD-system can be beneficial.

In *Use Case 2 – Virtual Design Review* we present the easy access to engineering data in a seamless and interactive workflow. With a *Live Link*, information created in Inventor can be reused in VRED in real-time. This example shows the control of visibility and transformation. So, variants defined as iLogic rules in Inventor can be easily brought to VRED. Even complex animations - created in Inventor - can easily be visualized in VRED with this workflow.

This can reduce the preparation effort for design reviews or product presentations significantly. The meeting itself can be done in collaboration with other participants independent from locations. With the transfer of the documented decisions back into Inventor the loop is closed

Live Link

The idea of this live link between Autodesk Inventor and Autodesk VRED is based on using an exchange file to pass information from one application to another (see Figure 6).

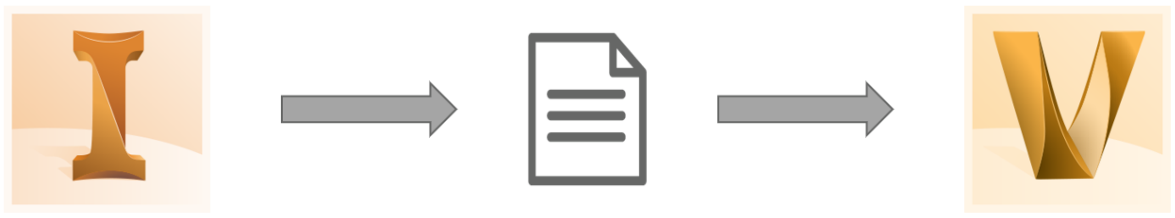
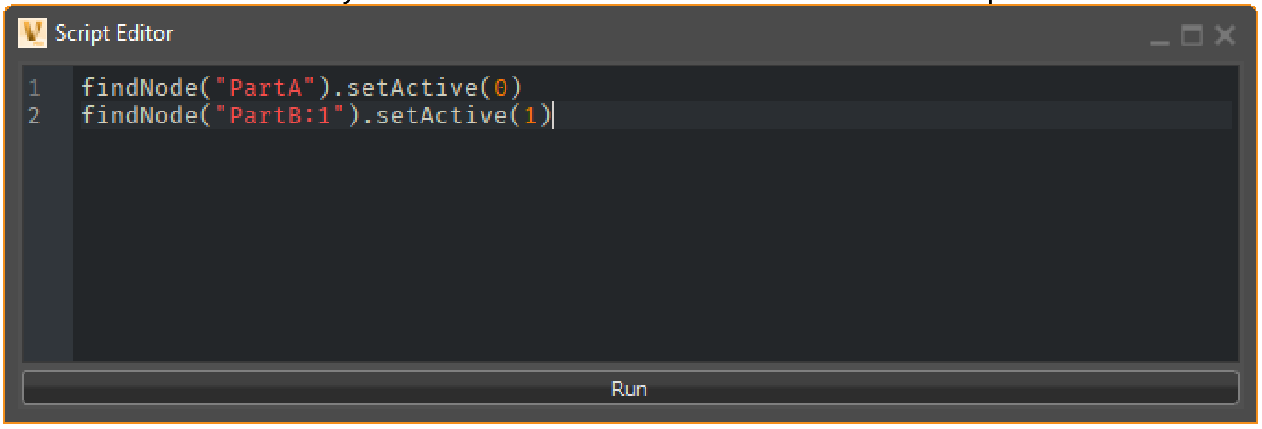


FIGURE 6

In Autodesk VRED visibility or transformation can be controlled with the Script Editor.



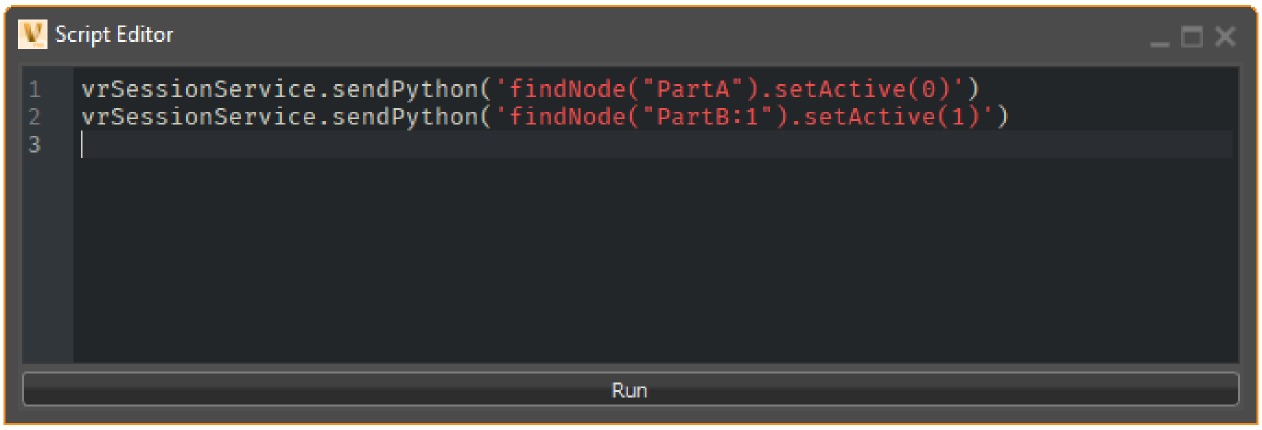
```
findNode("PartA").setActive(0)
findNode("PartB:1").setActive(1)
```

FIGURE 7

Figure 7 shows how the visibility of nodes (“PartA” and “PartB:1”) is controlled in the Script Editor.

`setActive(0)` = invisible
`setActive(1)` = visible

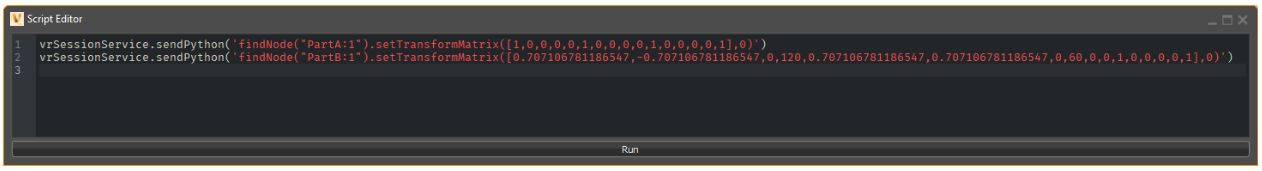
The script for a collaboration session is getting more complex (see Figure 8).



```
vrSessionService.sendPython('findNode("PartA").setActive(0)')
vrSessionService.sendPython('findNode("PartB:1").setActive(1)')
```

FIGURE 8

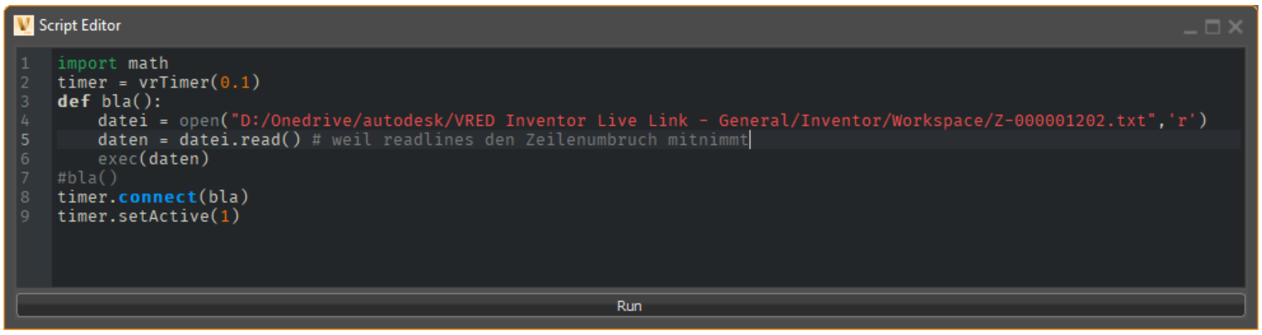
And for moving nodes/objects around the scene, the script gets very complex (see Figure 9).



```
vrSessionService.sendPython('findNode("PartA:1").setTransformMatrix([1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1],0)')
vrSessionService.sendPython('findNode("PartB:1").setTransformMatrix([0.707106781186547,-0.707106781186547,0,120,0.707106781186547,0.707106781186547,0,60,0,0,1,0,0,0,0,1],0)')
```

FIGURE 9

To automate the process of script input in the Editor, we decided to save the VRED script in an exchange file and use a Python script to load the exchange file into VRED (see Figure 10)



```
import math
timer = vrTimer(0.1)
def bla():
    datei = open("D:/Onedrive/autodesk/VRED Inventor Live Link - General/Inventor/Workspace/Z-000001202.txt",'r')
    daten = datei.read() # weil readlines den Zeilenumbruch mitnimmt
    exec(daten)
#bla()
timer.connect(bla)
timer.setActive(1)
```

FIGURE 10

The Script 1 in Figure 10 reads the exchange file “Z-000001202.txt” with an interval of 0.1 second defined by: `timer = vrTimer(0.1)`

```
import math
timer = vrTimer(0.1)
def bla():
    datei = open("PATH/EXCHANGE.txt",'r')
    daten = datei.read() # weil readlines den Zeilenumbruch mitnimmt
    exec(daten)
#bla()
timer.connect(bla)
timer.setActive(1)
```

SCRIPT 1

To create the exchange file, we used Autodesk Inventor.

We needed to capture the visibility and transformation of components in Autodesk Inventor.

Transformation Matrix

Both applications are using transformation matrices to describe the transformation of objects in the model space. Let's take a quick look on how the transformation matrices look like in Figure 11.

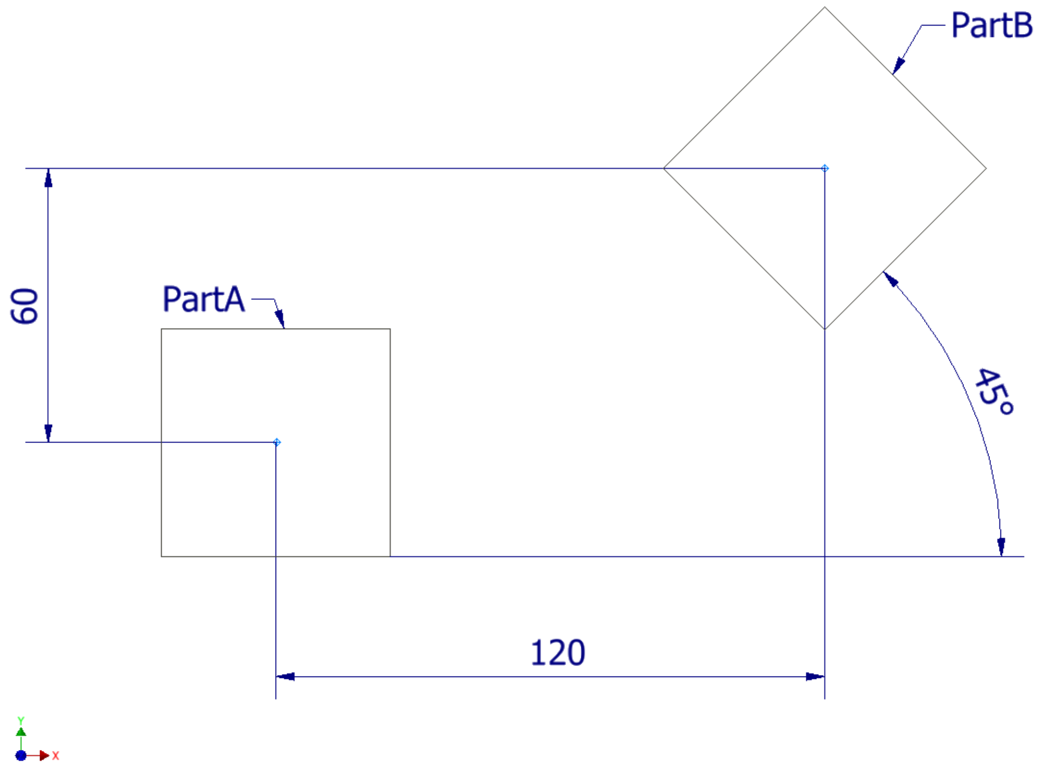


FIGURE 11

PartA is located in the origin of the global coordinate system. There is no positional change or rotation applied to it. Figure 12 shows the transformation matrix of PartA. The transformation matrix for that part consists of 4 parts.

Vector X direction = (1, 0, 0)
Vector Y direction = (0, 1, 0)
Vector Z direction = (0, 0 ,1)
Transformation Vector = (0, 0, 0)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FIGURE 12

PartB is moved in X direction by 120, in Y direction by 60 and rotated around Z axis by 45 degrees. The transformation matrix for PartB will look like shown in Figure 13.

$$\begin{bmatrix} 0.707 & -0,707 & 0 & 120 \\ 0.707 & 0.707 & 0 & 60 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FIGURE 13

By capturing the transformation matrix, we are able to know the exact location and rotation of a component in space.
If you want to learn more about transformation matrixes check “How Deep is the Rabbit Hole? Examing the Matrix and other Inventor® Math and Geometry Objects “ by Brian Ekins

From Inventor we need to pass **component name** and **transformation matrix** and put it into command that could be processed by VRED.
Example:
`vrSessionService.sendPython(‘findNode(“PartA:1”).setTransformMatrix([1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1],0)’)`

Script 2 creates the exchange file for each “PartA” and “PartB”.

```
Sub Main Save_posistion_of_given_Occurrences()

InventorVb.DocumentUpdate()

Dim oDoc As AssemblyDocument
oDoc = ThisApplication.ActiveDocument

Dim oDef As AssemblyComponentDefinition
oDef = oDoc.ComponentDefinition

'Open text file for writing. The text file will be placed in the main assembly
location
Dim oWrite As System.IO.TextWriter
oWrite = System.IO.File.CreateText(ThisDoc.PathAndFileName(False) & ".txt")

Dim oMatrix As Matrix

oOcc= Component.InventorComponent("PartA:1")
'Generate VRED script for “PartA:1”
Call SaveComponentMatrixInfo(oOcc, oWrite)

'Generate VRED script for “PartB:1”
oOcc = Component.InventorComponent("PartB:1")
Call SaveComponentMatrixInfo(oOcc, oWrite)

'Close file
oWrite.Close

InventorVb.DocumentUpdate()

End Sub


Function SaveComponentMatrixInfo(oOcc As ComponentOccurrence, oWrite As
System.IO.TextWriter)

   'Store occurrence name in CompName
   CompName = oOcc.Name
   'Read local transformation from component
   oMatrix = oOcc.Transformation

      Dim i As Integer
```

```
43        Mx = ""
44        'Read all 16 values of transformation matrix and store it in Mx string
45        For i = 1 To 4
46
47           Mx = Mx & oMatrix.Cell(i, 1) & ","
48           Mx = Mx & oMatrix.Cell(i, 2) & ","
49           Mx = Mx & oMatrix.Cell(i, 3) & ","
50
51           'The values in fourth raw in transformation matrix are related to scale
52           and are unitless
53                    If i <> 4 Then
54
55                            Mx = Mx & oMatrix.Cell(i, 4) * 10 & ","
56                    Else
57                              Mx = Mx & oMatrix.Cell(i, 4) & ","
58                    End If
59           Next
60
61           'Add "[" and "]" at start and end of the string and remove "," from the
62           end of the string to get format like this:
63           '[-0.7071,-0.7071,0,120,-0.7071,0.7071,0,60,0,0,-1,0,0,0,0,1]
64           Mx = "[" & Left(Mx, Len(Mx) -1) & "]"
65
66           'Add additional strings to get command read to use in VRED
67           'Example for PartB from handout:
68           vrSessionService.sendPython('findNode("PartB:1").setTransformMatrix([0.70
69           7,-0.707,0,120,0.707,0.707,0,60,0,0,1,0,0,0,0,1],0)')
70
71           TransformationList = "vrSessionService.sendPython('findNode(" & Chr(34) &
72           CompName & Chr(34) & ").setTransformMatrix(" & Mx & ",0)')"
73
74           'Write komponent transformation
75           oWrite.WriteLine(TransformationList)
76
77  End Function
```

SCRIPT 2

Step by step Instruction to setup live link for Transformation Matrix

By following the step by step instructions below you confirme that you have read, understood, and accepted the disclaimer.

> **DISCLAIMER:**
> In any case, all binaries, configuration code, templates and snippets of this solution are of "work in progress" character.
> Neither Przemek Sokolowski, nor Autodesk represents that these samples are reliable, accurate, complete, or otherwise valid.
> Accordingly, those configuration samples are provided “as is” with no warranty of any kind and you use the applications at your own risk.

1. Copy & Paste *Script 2* in a new iLogic rule in Inventor
2. Replace *PartA* and *PartB* in line 18 and 23 with a valid node name
   If you want to exchange more transformation matrix, copy and paste lines 18 to 20 for every component you want to exchange the information.
3. Run this rule to generate the exchange file
   This will be a TXT file with the same name as your assembly in the same location.
4. Copy & Paste *Script 1* in Script Editor in VRED
5. Replace *PATH* in line 4 with the path to the file location
   Example: D:/ProjectA/DesignB
   Important: Use / instead of \ to describe file location

6. Replace *EXCHANGE* with the filename of your exchange file created in step 3.
7. Change position of the components from step 2 and run the rule. Check if VRED updated change of components position

Please download the provided ZIP file from AU Class Site to get a detailed tutorial with sample dataset and video showing the steps.

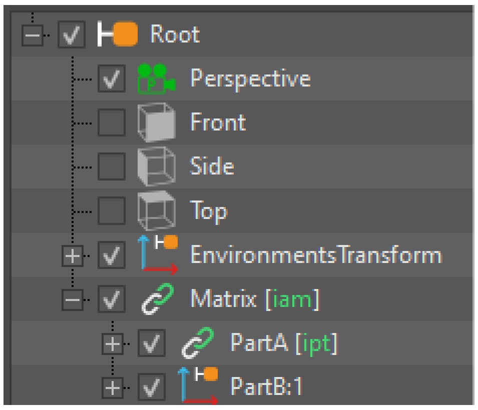
Visibility

To pass visibility of compnent script looks different from transformation matrix.
Example:

```
vrSessionService.sendPython('findNode("PartA").setActive(0)')
vrSessionService.sendPython('findNode("PartB:1").setActive(1)')
```

Visibility is controlled by setting 0 or 1 in setActive()

Please note that when importing a scene to VRED different node names are applied by VRED. If the component has no transformation or rotation applied, VRED is creating a node with the name that is equal to file name of the Inventor component (PartA.ipt). If there is a transformation applied, VRED is using the component name from Inventor browser (PartB:1). In Figure 14 the nodes of VRED (left) and Inventor (right) are directly compared.



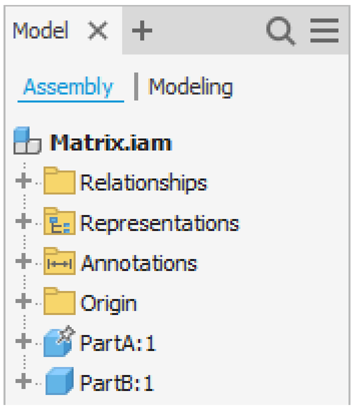


FIGURE 14

This must be considered when writing the script. The code is very similar to Script 2. The Script 3 checks the transformation matrix. If there is no transformation then the file name is used to address the node in VRED:

```
Sub Main Save_visibility_of_root_Occurrences()

InventorVb.DocumentUpdate()

Dim oDoc As AssemblyDocument
oDoc = ThisApplication.ActiveDocument

Dim oDef As AssemblyComponentDefinition
oDef = oDoc.ComponentDefinition

'Open text file for writing
Dim oWrite As System.IO.TextWriter
oWrite = System.IO.File.CreateText(ThisDoc.PathAndFileName(False) & ".txt")

Dim oMatrix As Matrix

'Browse through occurrences
Dim oOcc As ComponentOccurrence
```

```

'Go through all root occurrences
For Each oOcc In oDef.Occurrences
        Call SaveComponentVisibilityInfo(oOcc, oWrite)
Next

'Close file
oWrite.Close

InventorVb.DocumentUpdate()

End Sub



Function SaveComponentVisibilityInfo(oOcc As ComponentOccurrence, oWrite As
System.IO.TextWriter)

        Dim CompName As String
        CompName = oOcc.Name
        oMatrix = oOcc.Transformation

        Dim i As Integer
        Mx = ""
        For i = 1 To 4
                Mx= Mx & oMatrix.Cell(i, 1) & ","
                Mx= Mx & oMatrix.Cell(i, 2) & ","
                Mx = Mx & oMatrix.Cell(i, 3) & ","

                If i<>4 Then
                        Mx = Mx & oMatrix.Cell(i, 4) * 10 & ","
                Else
                        Mx = Mx & oMatrix.Cell(i, 4) & ","
                End If
        Next

        'Add "[" and "]" at start and end of the string
        'and remove "," from the end of the string
        Mx = "[" & Left(Mx, Len(Mx) -1) & "]"

        'Check if transformation exist. Without transformation VRED
        'is using file name of the component
        If Mx = "[1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1]" Then
                'Extract VRED nodename as file name
                CompName = oOcc.ReferencedDocumentDescriptor.FullDocumentName
                CompName = CompName.Split("\").Last()
                CompName = Left(CompName, Len(CompName)-4)
        End If

        'Component visibility
        If oOcc.Visible Then
                vis = 1
        Else
                vis = 0
        End If

        Visibility = "vrSessionService.sendPython('findNode(" & Chr(34) _
        & CompName & Chr(34) & ").setActive(" & vis & ")')"

        'Write komponent transformation
        oWrite.WriteLine(Visibility)

End Function
```

SCRIPT 3

Step by step Instruction to setup live link for Visibility

By following the step by step instructions below you confirme that you have read, understood, and accepted the disclaimer.

> **DISCLAIMER:**
> In any case, all binaries, configuration code, templates and snippets of this solution are of "work in progress" character.
> Neither Przemek Sokolowski, nor Autodesk represents that these samples are reliable, accurate, complete, or otherwise valid.
> Accordingly, those configuration samples are provided “as is” with no warranty of any kind and you use the applications at your own risk.

1. Copy & Paste Script 3 in a new iLogic rule in Inventor
   This rule only applies to first level components.
2. Run this rule to generate the exchange file
   This will be a TXT file with the same name as your assembly in the same location.
3. Copy & Paste Script 1 in Script Editor in VRED
4. Replace PATH in line 4 with the path to the file location
   Example: D:/ProjectA/DesignB
   Important: Use / instead of \ to describe file location
5. Replace EXCHANGE with the filename of your exchangefile created in step 3.
6. After each change of component visibility run the rule and see updated scene in VRED

Please download the provided ZIP file from AU Class Site to get a detailed tutorial with sample dataset and video showing the steps.

Collaboration

VRED provides an out-of-the-box collaboration solution for both desktop and VR users alike. Use it for showing work to a group in VR. Participants can join a collaboration session from their desktop or using HMDs.

Learn more about collaboration with VRED on Autodesk Knowledge Network

In our example, three people from three different locations, based in their home offices, met to collaborate in a virtual environment. The number of participants is not limited.

The avatars show the position of other participants in a scene. Details can be highlighted by a pointer. Decisions can be documented in annotations.

As it can be difficult to take notes while using an HMD, annotations can also be created by using voice recognition.

Learn more about Python 3 based Voice Recognition in VRED

Annotation

The annotation from VRED can be saved in xml format. To move it to Autodesk Inventor we need to extract annotation texts and its location coordinates.

Script 4 shows an example xml file with annotations from VRED:

```
<?xml version="1.0"?>
<!DOCTYPE annotations>
<Annotations>
 <TextAnnotation name="Annotation" node="Solid1">
  <Text>I like the design. Apporoved!</Text>
  <Position x="1329.5" y="156.235" z="3842.36"/>
  <Visible>1</Visible>
  <UseNodeVisibility>0</UseNodeVisibility>
  <FontColor a="1" b="1" g="1" r="1"/>
  <LineColor a="1" b="0.00999466" g="0.44799" r="1"/>
  <BackColor a="0.6" b="0.0732433" g="0.0732433" r="0.0732433"/>
  <Size>0.4</Size>
  <Scale far="2000" near="800"/>
 </TextAnnotation>
 <TextAnnotation name="Annotation1" node="Volumenkörper999">
  <Text>Simon: This is beautiful Steel. :)</Text>
  <Position x="-289.09" y="2870.76" z="4461.81"/>
  <Visible>1</Visible>
  <UseNodeVisibility>0</UseNodeVisibility>
  <FontColor a="1" b="1" g="1" r="1"/>
  <LineColor a="1" b="0.00999466" g="0.44799" r="1"/>
  <BackColor a="0.6" b="0.0732433" g="0.0732433" r="0.0732433"/>
  <Size>0.4</Size>
  <Scale far="2000" near="800"/>
 </TextAnnotation>
 <TextAnnotation name="Annotation2" node="Volumenkörper227">
  <Text>nice forklift animation!</Text>
  <Position x="910.082" y="15192" z="2189.14"/>
  <Visible>1</Visible>
  <UseNodeVisibility>0</UseNodeVisibility>
  <FontColor a="1" b="1" g="1" r="1"/>
  <LineColor a="1" b="0.00999466" g="0.44799" r="1"/>
  <BackColor a="0.6" b="0.0732433" g="0.0732433" r="0.0732433"/>
  <Size>0.4</Size>
  <Scale far="2000" near="800"/>
 </TextAnnotation>
</Annotations>
```

SCRIPT 4

The iLogic rule in Script 5 reads the xml file and creates 3D Annotations in Autodesk Inventor:

```
AddReference "System.Linq"
AddReference "System.Xml"
AddReference "System.Xml.Linq"
AddReference "System.Core"
Imports System.Linq
Imports System.Xml
Imports System.Xml.Linq
Imports System.Xml.Schema

Class FunctionalClass
    Shared FeedXML As XDocument

    Sub Main()

        Dim oFileDlg As Inventor.FileDialog = Nothing
        InventorVb.Application.CreateFileDialog(oFileDlg)
        Dim FileList() As String
        oFileDlg.Filter = "XML Files(*.xml)|*.xml|All Files (*.*)|*.*"

        oFileDlg.InitialDirectory = ThisDoc.Path
        'oFileDlg.InitialDirectory ="C:\temp\"
        oFileDlg.CancelError = True
        On Error Resume Next
        oFileDlg.MultiSelectEnabled=False
        oFileDlg.ShowOpen()

        If Err.Number <> 0 Then
                MessageBox.Show("File not chosen.", "Dialog Cancellation")
        ElseIf oFileDlg.FileName <> "" Then
                FileList = oFileDlg.FileName.Split("|")
        End If

        FeedXML = XDocument.Load(FileList(0))

        Dim MyTextArray As ArrayList = New ArrayList()
        Dim Note As String

        Dim MyPositionArray As ArrayList = New ArrayList()

        'Create the list of all text annotations and store it in MyTextArray

For Each oElement As XElement In FeedXML.Descendants("Text")
                Note=oElement
                MyTextArray.Add(Note)
        Next

        'Create the list of all text annotations coordinates
For Each oElement As XAttribute In FeedXML.Descendants("Position").Attributes
                Note = oElement
                MyPositionArray.Add(Note)
        Next

        j=0
        'Create sets of text annotation (Note) and corresponding coordinates
        X,Y,Z
        For i = 0 To MyTextArray.Count - 1
                Note = MyTextArray(i)
                X = Val(MyPositionArray(j)) / 10
                Y = Val(MyPositionArray(j + 1)) / 10

                Z = Val(MyPositionArray(j + 2)) / 10

                j = j + 3

                'Create 3D annotation in Inventor
                Call Create3dAnnotation(Note, X, Y, Z)
        Next i
        End Sub

Function Create3dAnnotation(Note As String, X As Double, Y As Double, Z As
Double)
        'Add MBD annotation
        Dim Doc As AssemblyDocument
        Doc = ThisApplication.ActiveDocument

        Dim oDef As AssemblyComponentDefinition
        oDef = Doc.ComponentDefinition

        Dim oTG As TransientGeometry
        oTG = ThisApplication.TransientGeometry

```

```
 82         'Define point to attach annotation
 83         Dim WP As WorkPoint
 84         WP = oDef.WorkPoints.AddFixed(oTG.CreatePoint(X,Y,Z))
 85 
 86         Dim dViewRepMgr As RepresentationsManager
 87         dViewRepMgr = oDef.RepresentationsManager
 88 
 89         Dim dWeldView As DesignViewRepresentations
 90         dWeldView = dViewRepMgr.DesignViewRepresentations
 91 
 92         'Define plane for placing annotation (Global ZX)
 93         Dim oAnnoPlaneDef As AnnotationPlaneDefinition
 94         oAnnoPlaneDef =
 95         oDef.ModelAnnotations.CreateAnnotationPlaneDefinitionUsingPlane(oDef.Work
 96         Planes.Item(2))
 97 
 98         Dim oLeaderPoints As ObjectCollection
 99         oLeaderPoints = ThisApplication.TransientObjects.CreateObjectCollection
100         oLeaderPoints.Add(oTG.CreatePoint(X + 500, Y + 100, Z + 250))
101 
102         Dim oLeaderIntent As GeometryIntent
103         oLeaderIntent = oDef.CreateGeometryIntent(WP)
104         oLeaderPoints.Add(oLeaderIntent)
105 
106         Dim oLeaderDef As ModelLeaderNoteDefinition
107         oLeaderDef =
108         oDef.ModelAnnotations.ModelLeaderNotes.CreateDefinition(oLeaderPoints,
109         Note, oAnnoPlaneDef)
110         Dim oLeader As ModelLeaderNote
111         oLeader= oDef.ModelAnnotations.ModelLeaderNotes.Add(oLeaderDef)
112 
113 End Function
114 
115 End Class
```

SCRIPT 5

Step by step Instruction to Import Annotations to Inventor

By following the step by step instructions below you confirme that you have read, understood, and accepted the disclaimer.

DISCLAIMER:
In any case, all binaries, configuration code, templates and snippets of this solution are of "work in progress" character.
Neither Przemek Sokolowski, nor Autodesk represents that these samples are reliable, accurate, complete, or otherwise valid.
Accordingly, those configuration samples are provided “as is” with no warranty of any kind and you use the applications at your own risk.

1. Save annotations as XML file in VRED
2. Go to your open Inventor file
3. Copy & Paste Script 5 in a new iLogic rule in Inventor
4. Replace C:\temp\ in line 21 with the valith path to your exported XML file in step 1
   Important: Use \ instead of / to describe file location
5. Run this rule to create the 3D Annotations in your Inventor file

Please download the provided ZIP file from AU Class Site to get a detailed tutorial with sample dataset and video showing the steps.

Conclusion

High quality visualization is key for a compelling product presentation. The way in which decisions are made is changing through digitalization. To save costs and time, reality is replicated for quick design exploration and presentation. Through visual communication and collaboration, design intent and quality can be better understood. A good visualization gives confidence in digital decision making.

By following four simple steps, the engineering data is ready to be used for sales, marketing and decision making.

1. Import design data
2. Adjust Materials and Look
3. Create a compelling storyboard
4. Define Output

Take advantage of the scripting capabilities of Inventor and VRED to directly reuse information from engineering data. Different variants can be easily evaluated during a design review by synchronizing multiple tools in multiple locations with multiple devices. This is the future of digital decision making.