Assembly Document - Essentials

Presenter
Developer Technical Services

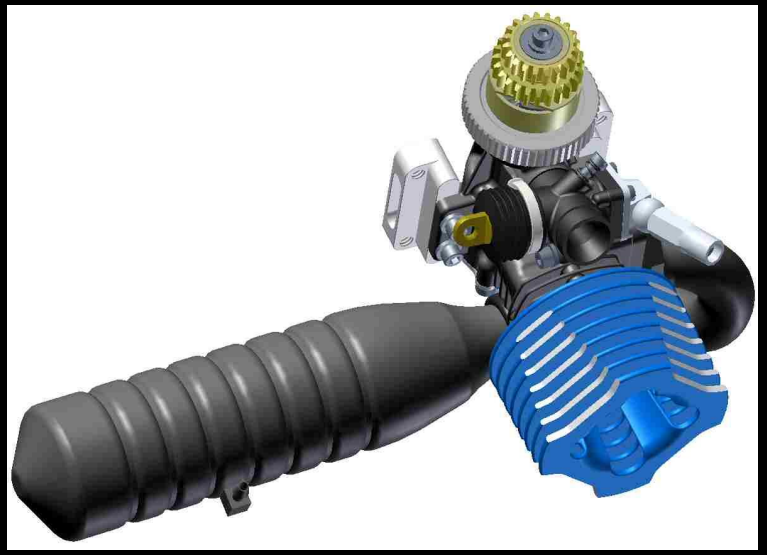
Agenda

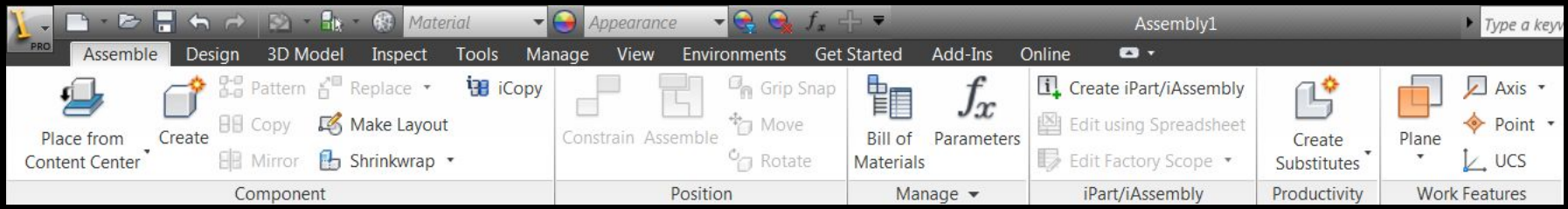
- Assembly End User vs API
- Assembly Structure
- Transient Geometry: using Matrices and Vectors
- Proxies
- Constraints
- Lab: Constraints creation

Autodesk®

Assembly Documents

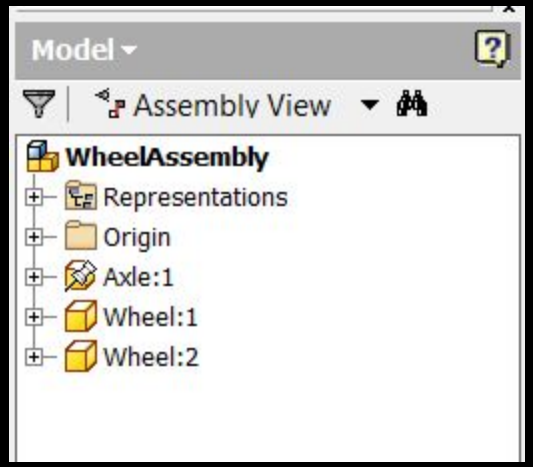
- The API supports most of the assembly functionality.
  - Placing & creating components.
  - Editing components
  - Patterns
  - Constraints
  - Work features
  - Parameters
  - iMates
  - Sketches
  - Features
  - Representations
  - iAssemblies
  - BOM

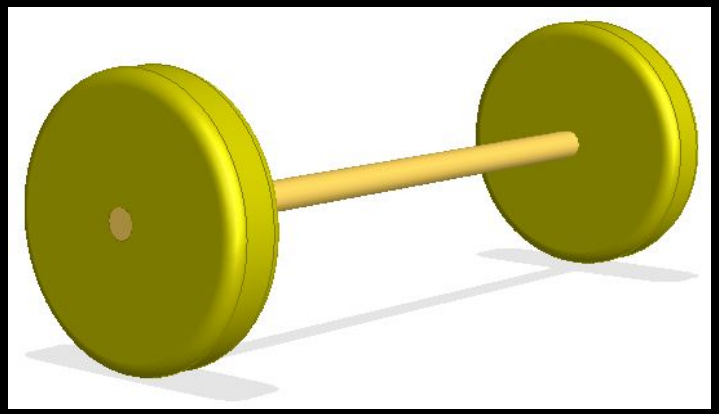




Autodesk

Assembly Document as an End User





Autodesk

Assembly Document Through the API

- Assembly documents contain:
  - references to other documents
  - occurrence information, constraints
  - work features
- No geometry is in the assembly document, only references to parts and other assemblies. (Assembly features are a special case exception.)

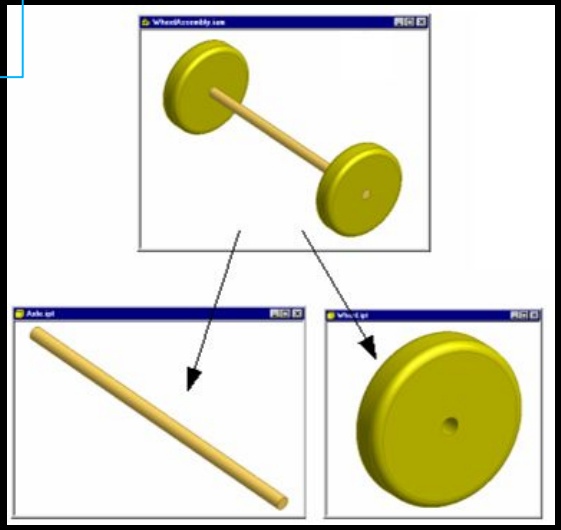
WheelAssembly.iam

References:

1. Axle.ipt
2. Wheel.ipt

Occurrences:

1. Axle:1, Reference1, (0,0,0,…), Visible, …
2. Wheel:1, Reference2, (0,0,-2,…), Visible, …
3. Wheel:2 Reference3, (0,0,-2,…), Visible, …



Autodesk

Assembly Document Structure API

- The ***ComponentOccurrences*** object is accessed through the ***Occurrences*** property and allows iteration over all existing occurrences and provides support to add additional occurrences.
- The ***DocumentDescriptorsEnumerator*** object is accessed through the ***ReferencedDocumentDescriptors*** property and provides access to the documents referenced by this document.

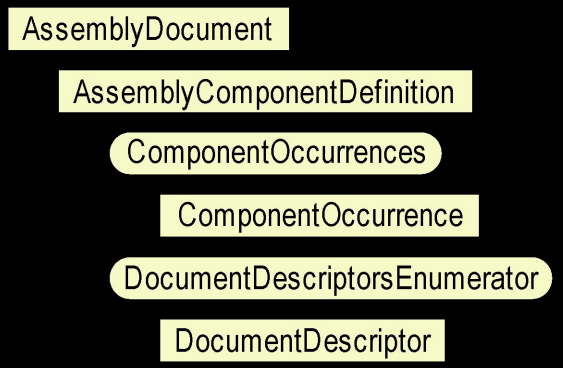
WheelAssembly.iam

References:

1. Axle.ipt
2. Wheel.ipt

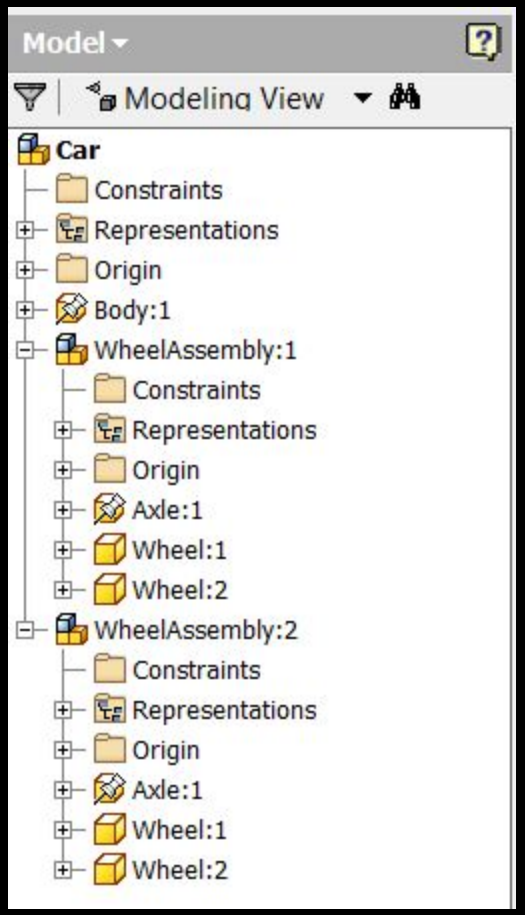
Occurrences:

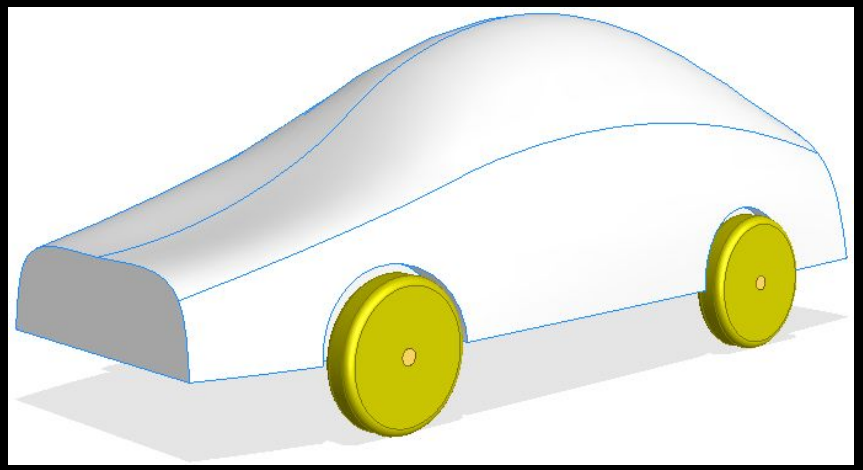
1. Axle:1, Reference1, (0,0,0,…), Visible, …
2. Wheel:1, Reference2, (0,0,-2,…), Visible, …
3. Wheel:2 Reference3, (0,0,-2,…), Visible, …



Autodesk®

Assembly Structure Traversal





AssemblyDocument

AssemblyComponentDefinition

ComponentOccurrences

ComponentOccurrence

ComponentOccurrencesEnumerator

ComponentOccurrence

Autodesk

Assembly Structure Traversal - Example

```
Public Sub AssemblyTraversal()

    ' Get the active document, assuming it's an assembly.
    Dim oAsmDoc As AssemblyDocument
    oAsmDoc = _InvApplication.ActiveDocument

    Call TraverseAsm(oAsmDoc.ComponentDefinition.Occurrences, 1)

End Sub

Private Sub TraverseAsm(ByVal oOccurrences As ComponentOccurrences, ByVal Level As Integer)

    ' Iterate through the current list of occurrences.
    Dim oOcc As ComponentOccurrence
    For Each oOcc In oOccurrences
        ' Print the name of the current occurence.
        Debug.Print(Space(Level * 3) & oOcc.Name)

        ' If the current occurrence is a subassembly then call this sub
        ' again passing in the collection for the current occurrence.
        If oOcc.DefinitionDocumentType = DocumentTypeEnum.kAssemblyDocumentObject Then
            Call TraverseAsm(oOcc.SubOccurrences, Level + 1)
        End If
    Next

End Sub
```


Autodesk®

Creating Occurrences

- ***Add***(FileName As String, Position As Matrix) As ComponentOccurrence
- ***AddByComponentDefinition***(CompDef As ComponentDefinition, Position As Matrix) As ComponentOccurrence
- ***AddUsingiMates***(FileName As String, Position As Matrix) As ComponentOccurrence
- ***AddCustomiPartMember***(FactoryFileName As String, Position As Matrix, FullFileName As String, [Row], [CustomInput]) As ComponentOccurrence
- ***AddiPartMember***(FactoryFileName As String, Position As Matrix, [Row]) As ComponentOccurrence
- ***AddiAssemblyMember***(FactoryDocumentName As String, Position As Matrix, [Row], [Options]) As ComponentOccurrence

Autodesk®

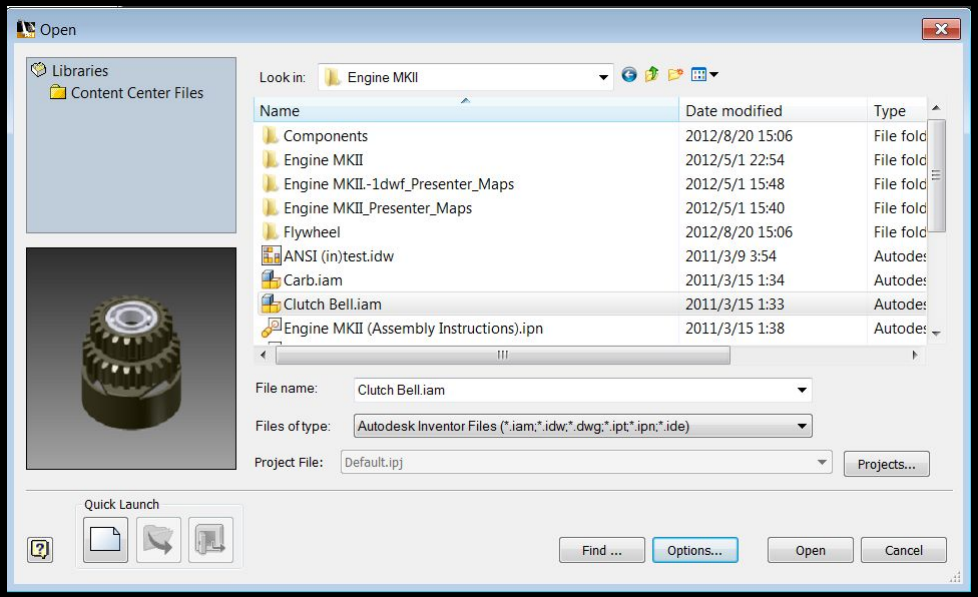
Creating an Occurrence - Example

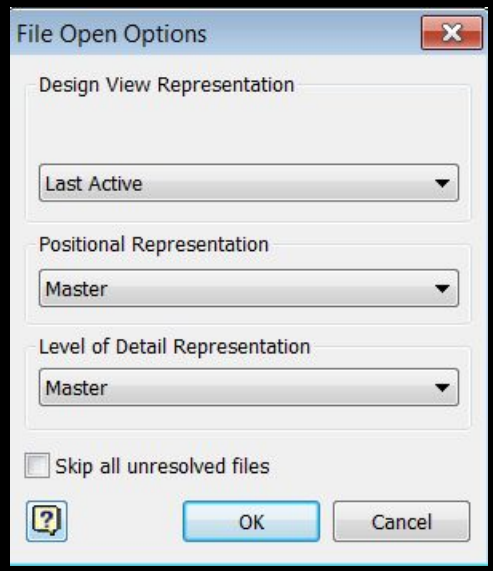
```
Public Sub AddFromFile()

    Dim oDoc As AssemblyDocument
    oDoc = _InvApplication.ActiveDocument

    Dim oMatrix As Matrix
    oMatrix = _InvApplication.TransientGeometry.CreateMatrix

    Dim oOcc As ComponentOccurrence
    oOcc = oDoc.ComponentDefinition.Occurrences.Add("C:\Temp\Part1.ipt", oMatrix)

End Sub


Public Sub AddFromMemory()

    Dim oDoc As AssemblyDocument
    oDoc = _InvApplication.ActiveDocument

    Dim oPartDoc As PartDocument
    oPartDoc = _InvApplication.Documents.Add(kPartDocumentObject, False)

    Dim oMatrix As Matrix
    oMatrix = _InvApplication.TransientGeometry.CreateMatrix

    Dim oOcc As ComponentOccurrence
    oOcc = oDoc.ComponentDefinition.Occurrences.AddByComponentDefinition( _
                    oPartDoc.ComponentDefinition, oMatrix)

End Sub
```


Autodesk®

Creating Occurrences With Options

- **AddWithOptions(FullDocumentName As String, Position As Matrix, _ Options As NameValueMap) As ComponentOccurrence**
- Options
  - PrivateRepresentationFileName
  - DesignViewRepresentation
  - PositionalRepresentation
  - LevelOfDetailRepresentation
  - UseiMate
  - DesignViewAssociative





Autodesk

Add With Options - Example

```
'Create a new NameValueMap object
Dim oOptions As NameValueMap
oOptions = _InvApplication.TransientObjects.CreateNameValueMap

' Set the representations to use when creating the occurrence.
Call oOptions.Add("LevelOfDetailRepresentation", "MyLODRep")
Call oOptions.Add("PositionalRepresentation", "MyPositionalRep")
Call oOptions.Add("DesignViewRepresentation", "MyDesignViewRep")
Call oOptions.Add("DesignViewAssociative", True)

' Add the occurrence.
Dim oOcc As ComponentOccurrence
oOcc = oAsmCompDef.Occurrences.AddWithOptions("C:\Temp\Reps.iam", _
                                              oMatrix, oOptions)
```


Autodesk®

Transient Geometry Math Objects

- The ***TransientGeometry*** object allows you to create some mathematical objects that can be used as input for methods and properties and also used internally for your own calculations.
  - Point, Point2d
  - Matrix, Matrix2d
  - Vector, Vector2d
  - UnitVector, UnitVector2d
  - Box, Box2d

Autodesk®

What is a Matrix?

- A matrix is a rectangular array of numbers.
- A 3-D matrix is a 4x4 matrix.

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

- A 2-D matrix is a 3x3 matrix.

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

Autodesk®

A Matrix in Inventor

- In computer graphics a matrix is commonly used to:
  - Define a coordinate system.
  - Define a transformation.
- Inventor uses this concept for occurrences in assemblies, sketches in parts, and drawing view contents transformations.

Autodesk®

Matrix and Occurrences

- When placing an occurrence the matrix defines the position of the part within the assembly. It defines the position of the part coordinate system within the assembly space.
- Getting the ***Transformation*** property of an occurrence returns the matrix that defines the occurrence's current position in the assembly.
- Setting the ***Transformation*** property repositions the occurrence (taking into account any constraints).
- ***SetTransformWithoutConstraints*** transforms the occurrence ignoring any constraints (until the next recompute of the assembly).

Autodesk®

Matrix as a Transform

- A matrix can be used to define a transformation for an existing object.
  - Repositioning an occurrence within an assembly.
  - Defining the change from one coordinate system to another. For example, in an assembly transforming a point from one part into another part.
- For a transformation the matrix defines the delta change to apply. The change can be a move and/or a rotate.

Autodesk®

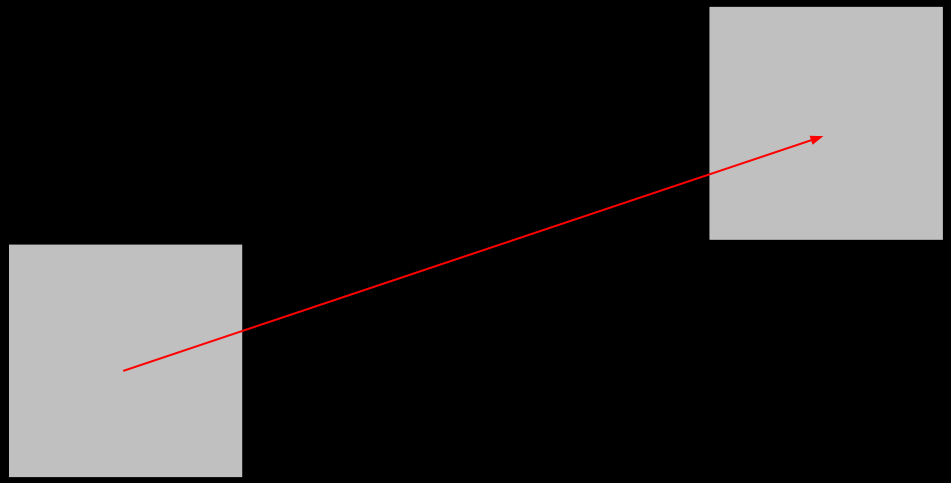
Matrix Functions

- ***Matrix.Invert*** reverses the transform the matrix defines.
- ***Matrix.TransformBy*** changes the matrix to include the transformation defined by a second matrix.
- ***Matrix.Cell*** allows you to get/set individual cells of the matrix.
- ***SetCoordinateSystem***, ***SetToAlignCoordinateSystems***, ***SetToIdentity***, ***SetToRotateTo***, ***SetToRotation***, *and* ***SetTranslation*** are for convenience in defining the matrix.

Autodesk®

Vectors

- Vectors define a direction and magnitude.
- A Vector can be used to define the movement of the part shown below.
- A UnitVector defines a direction. Its magnitude is always 1.



Autodesk

Lab: Positionning Occurrences

- Write a .Net program with 2 methods:
  - 1. A method that creates an assembly document, inserts an occurrence in the new assembly with no specific transformation
  - 2. A method that takes as input:
    - an occurrence
    - a translation vector Tx
    - an axis vector Ax
    - an angle Alpha (in degrees)

    And that translates the occurrence of Tx, rotates it of Alpha around axis Ax with the center of rotation at the occurrence gravity center.

Autodesk

Assembly Document - Proxies

- Q: How do you access geometry within the context of an assembly since geometry doesn't exist in assemblies?
- A: A proxy represents an entity as if the entity actually exists in the assembly.

AssemblyDocument

AssemblyComponentDefinition

ComponentOccurrences

ComponentOccurrence

PartDocument AssemblyDocument

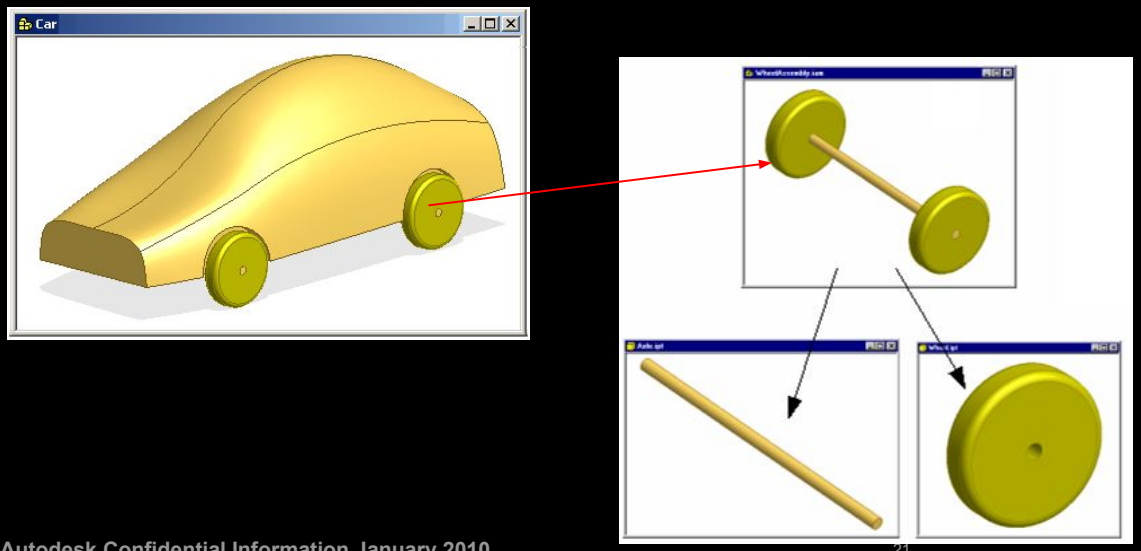
ComponentDefinition

ComponentOccurrencesEnumerator

ComponentOccurrenceProxy

SurfaceBodies

SurfaceBodyProxy



Autodesk Confidential Information January 2010

Autodesk

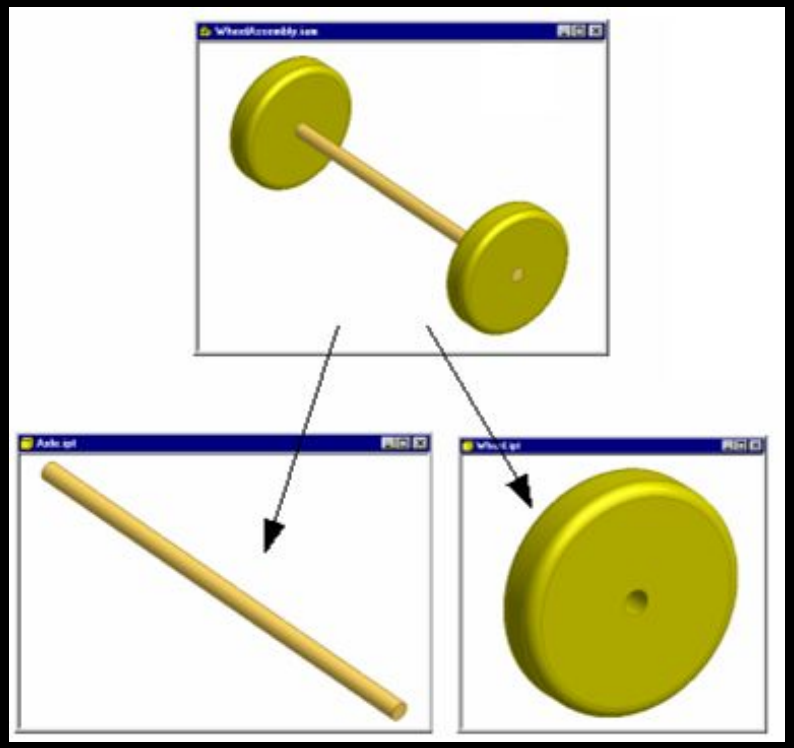
Proxy Objects

- Proxy objects are derived from the regular object they represent.
  - They support every method and property the original object supports.
  - These methods and properties will return information in the context of the assembly.
- In addition to the functions of the base class object, proxies also support:
  - ***ContainingOccurrence*** – Returns the occurrence the proxy is representing the real object within.
  - ***NativeObject*** – Returns the actual object the proxy is representing.

Autodesk®

Proxy Objects

- Proxies define a path to the actual object.
  - Cylindrical Face 1
    Wheel:1\CylinderFace
  - Cylindrical Face 2
    Wheel:2\CylinderFace
- Proxies are returned when the user selects entities.
- Proxies can be created using the ***CreateGeometryProxy*** method.
- Existing proxy paths can be trimmed using ***AdjustProxyContext*** method.
- Paths can be examined using ***OccurrencePath*** property.



Autodesk

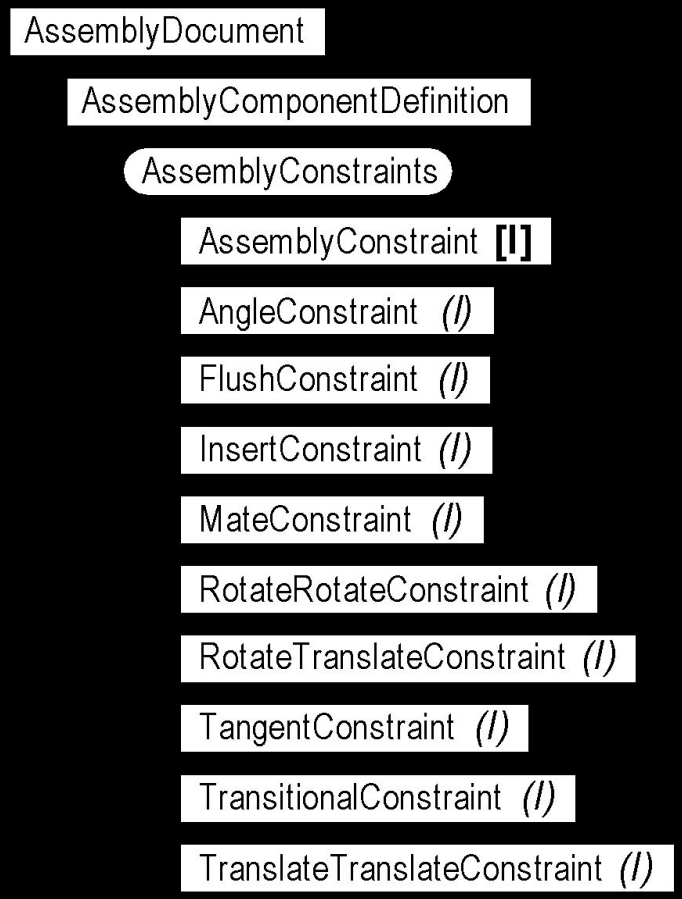
Creating Proxies - Example

```
Public Sub CreateProxy()

    Dim oAsmDef As AssemblyComponentDefinition
    oAsmDef = _InvApplication.ActiveDocument.ComponentDefinition

    Dim oOcc1 As ComponentOccurrence = oAsmDef.Occurrences(1)
    Dim oOcc2 As ComponentOccurrence = oAsmDef.Occurrences(2)

    ' Get the vertex through the occurrence
    ' which will return a VertexProxy object.
    Dim oVertexPx1 As VertexProxy
    oVertexPx1 = oOcc1.SurfaceBodies(1).Vertices(1)

    ' Get the vertex from the part and create a VertexProxy object.
    Dim oVertex2 As Vertex
    oVertex2 = oOcc2.Definition.SurfaceBodies(1).Vertices(1)

    Dim oVertexPx2 As VertexProxy = Nothing
    Call oOcc2.CreateGeometryProxy(oVertex2, oVertexPx2)

End Sub
```


Autodesk®

Assembly Document – Constraints

- Constraint creation can take as input work geometry from the assembly or proxies to entities in the attached parts.
- Query of a constraint returns the associated entities and the parameter controlling the constraint.



Autodesk

Adding Constraints – from native objects

```
Public Sub MateConstraintOfWorkPlanes()
 Dim oAsmCompDef As AssemblyComponentDefinition
  oAsmCompDef = ThisApplication.ActiveDocument.ComponentDefinition

 ' Get references to the two occurrences to constrain.
 ' This arbitrarily gets the first and second occurrence.
 Dim oOcc1 As ComponentOccurrence
   oOcc1 = oAsmCompDef.Occurrences.Item(1)

 Dim oOcc2 As ComponentOccurrence
   oOcc2 = oAsmCompDef.Occurrences.Item(2)

 ' Get the XY plane from each occurrence.  This goes to the
 ' component definition of the part to get this information.
 ' This is the same as accessing the part document directly.
 ' The work plane obtained is in the context of the part,
 ' not the assembly.
 Dim oPartPlane1 As WorkPlane
   oPartPlane1 = oOcc1.Definition.WorkPlanes.Item(3)

 Dim oPartPlane2 As WorkPlane
   oPartPlane2 = oOcc2.Definition.WorkPlanes.Item(3)

 ' Because we need the work plane in the context of the assembly
 ' we need to create proxies for the work planes.  The proxies
 ' represent the work planes in the context of the assembly.
 Dim oAsmPlane1 As WorkPlaneProxy
 Call oOcc1.CreateGeometryProxy(oPartPlane1, oAsmPlane1)

 Dim oAsmPlane2 As WorkPlaneProxy
 Call oOcc2.CreateGeometryProxy(oPartPlane2, oAsmPlane2)

 ' Create the constraint using the work plane proxies.
 Call oAsmCompDef.Constraints.AddMateConstraint(oAsmPlane1, oAsmPlane2, 0)
End Sub
```

Autode

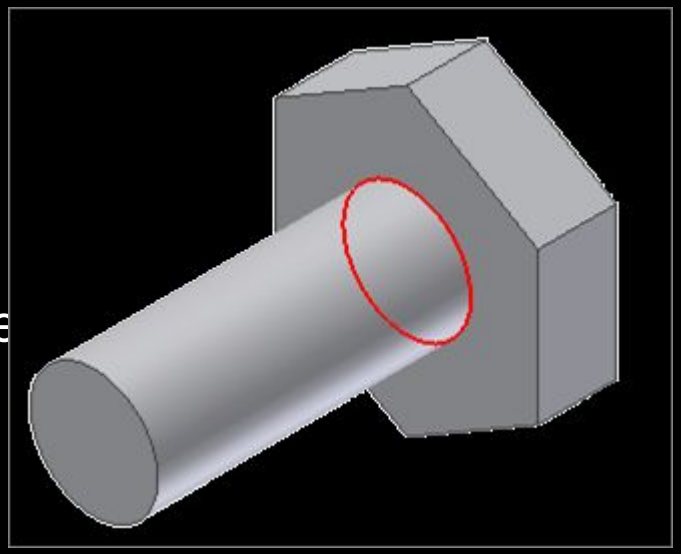
Autodesk

Adding Constraints – from proxy objects

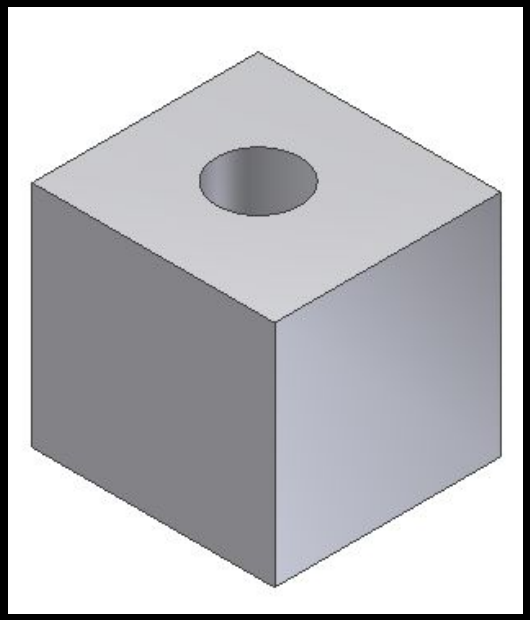
```
Public Sub MateConstraintWithLimits()

    ' Set a reference to the assembly component definintion.
    Dim oAsmCompDef As AssemblyComponentDefinition
      oAsmCompDef = ThisApplication.ActiveDocument.ComponentDefinition

    ' Set a reference to the select set.
    Dim oSelectSet As SelectSet
      oSelectSet = ThisApplication.ActiveDocument.SelectSet

    ' Validate the correct data is in the select set.
    If oSelectSet.Count <> 2 Then
        MsgBox ("You must select the two entities valid for mate.")
        Exit Sub
    End If

    ' Get the two entities from the select set.
    Dim oBrepEnt1 As Object
    Dim oBrepEnt2 As Object
      oBrepEnt1 = oSelectSet.Item(1)
      oBrepEnt2 = oSelectSet.Item(2)

    ' Create the mate constraint between the parts, with an offset value of 0.
    Dim oMate As MateConstraint
      oMate = oAsmCompDef.Constraints.AddMateConstraint(oBrepEnt1, oBrepEnt2, 0)

    ' Set a maximum value of 2 inches
    oMate.ConstraintLimits.MaximumEnabled = True
    oMate.ConstraintLimits.Maximum.Expression = "2 in"

    ' Set a minimum value of -2 inches
    oMate.ConstraintLimits.MinimumEnabled = True
    oMate.ConstraintLimits.Minimum.Expression = "-2 in"
End Sub
```


Autodesk®

Lab: Creation of constraints



1. Manually (not with the API), create a simple bolt part like the one shown to the right.
2. Write a program to add an attribute to the cylinder face. This is used to “name” the edge to allow you to find in the next program.
3. Create another part that’s a block with one blind hole, similar to the one shown to the right. Add an attribute to the face of the hole as well.
4. Write a program that will (with an assembly active)
   - Insert the block part into the assembly.
   - Insert a bolt part into the assembly.
   - Use the Attribute API to find the faces of the hole and the bolt face.
   - Create an insert constraint between the bolt and the block using the attribute on the bolt and the faces just found.



Autodesk

Autodesk®