AUTODESK UNIVERSITY

Advanced Inventor Modeling for Infraworks

TR500670

Danny Lewis
Hatch Global Lead – Autodesk
linkedin.com/in/dlewis145/

Special Thanks To Autodesk Team

Ara Ashekian, P. Eng.

Product Manager – Bridges and Civil Infrastructures

- Knows Infraworks like the back of his hand
- Will find the answer and/or fix the issue
- Got me in touch with the developers to provide a lot of the know how in this presentation

Kristopher M. Landry

Technical Solutions Executive – Hatch

- Introduced me to Ara 😜
- Finds the right people in Autodesk to ask the difficult questions to

Before we begin…

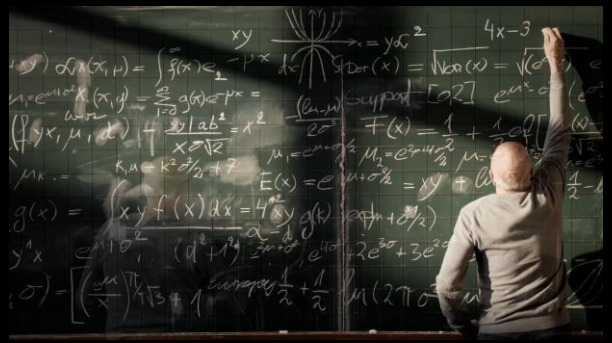
First, got to pat myself the back

AU 2021 Presentation

One of the Top-Rated Engineering Classes

Multibridge Cable Stay Intelligent Models in InfraWorks, Inventor, Civil 3D, and Revit

Scott Cameron, Danny Lewis



Agenda

What we’re going to learn today

Topics we'll cover

- Show the path for models from Inventor to InfraWorks
- Show creating simple toggles in Inventor
- Show the code for toggling lots of features at the same time
- Discuss the blue parameters vs. param()
- Discuss 'ThisApplication' vs. 'ThisServer'
- Demonstrate how to setup a Log File
- Discuss how to suppress things in parts and things in assemblies
- Demonstrate how to change part colours
- Demonstrate how to change text
- Demonstrate how to setup drop down lists
- Show "By Pattern" coding
  - SuperT example
- Setup Files from multi-body to assembly
  - Show off our HATCH developed tool to make this publishing easier (code not provided)

Why so much iLogic?

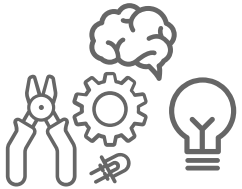
…because it's awesome

- What is iLogic and why does it matter when doing Infraworks models?

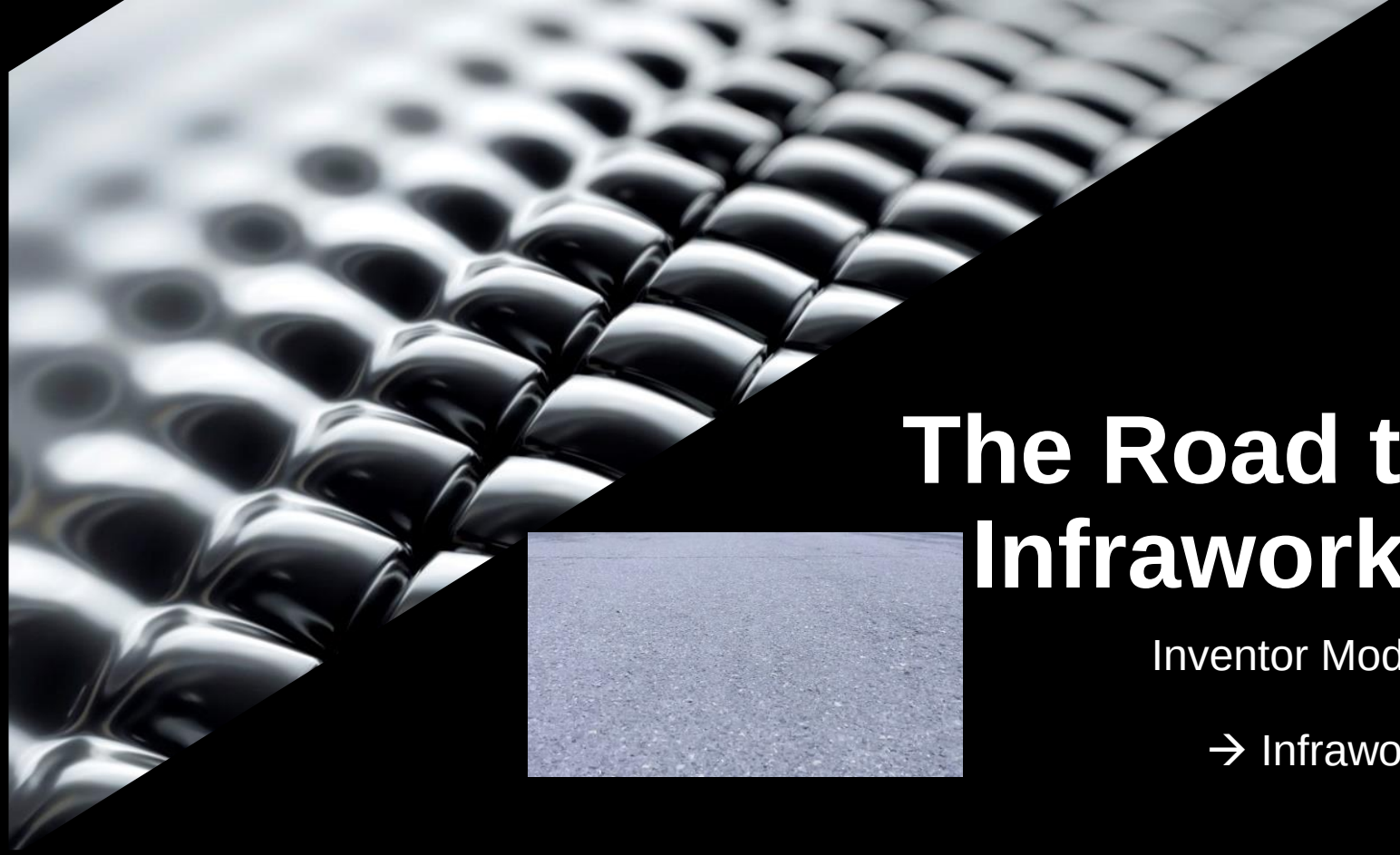
…time for some ranting

- Can edit and manipulate **any feature/part/thing/etc.** in Inventor via simple coding
- Once you connect toggles to the Inventor features, you can **interact and manipulate them in Infraworks**
- **Options are almost limitless** for what you can do once you start tapping into the coding side of Inventor

In Summary



Incredibly Versatile and Powerful Tool (better than Dynamo!)



The Road to Infraworks

Inventor Models

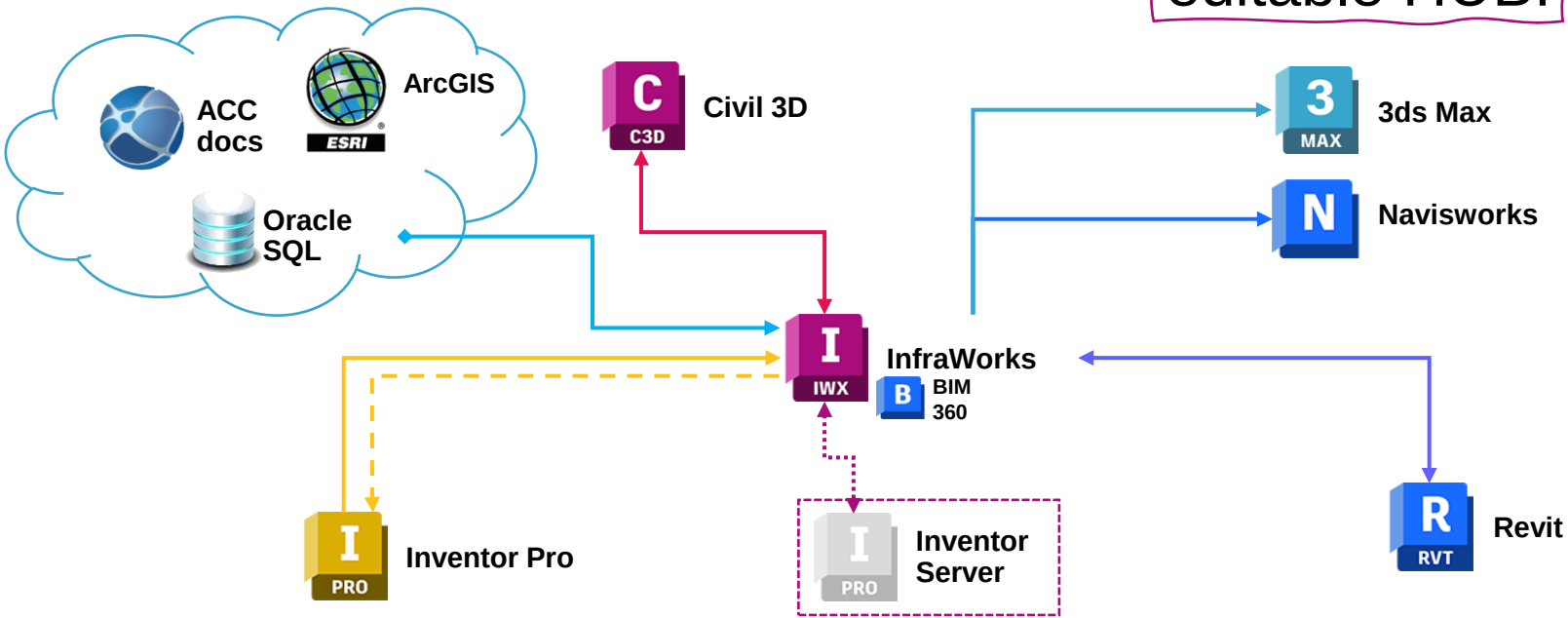
→ Infraworks

Inventor to Infraworks

…it never left Inventor!

- Infraworks is a very unique software in the Autodesk suite because:

It is an editable HUB!



Inventor to Infraworks

Likely outdated information

Behind the curtain

- Infraworks requires a number of things before it will '**accept**' an Inventor Model
  - All of these are just to help the system validate that you are giving it a correct model
  - Even if it's not perfect… it's 10,000x better than the garbage system for publishing Inventor models into Civil 3D

Infraworks will look for:

- .ipt or .iam — *Your Inventor Files*
- .xml file — *Describes what your files have in them*
- 2x .jpegs — *Pretty icon pictures*

Bonus:

- .json — *All the parameter information, sizes information, and data*

Infraworks will create:

- NEW .ipt or .iam — *Geom file & friends*
- .ACItem — *Basically same as the JSON file*
- .json (if you export one)

How to Bring in Inventor Parts

Wrong AU Presentation

- The specific details have been done before:

Inventor Pro → InfraWorks

See this class for those details

Autodesk is also making things better all the time…

Anything here would probably be out of date.

Ways it's probably better:

- Don't need 'the add-in' anymore
- Infraworks lets you 'refresh' your models (and remembers your specific instance parameters)
- Auto-generates the jpeg images (with a better background than the original add-in)
- Fixes the things that were broken on the add-in
- Custom property mapping to Infraworks driven things

Maybe they even fixed:

- Importing textures for girders (barely works)
- Better system for putting in the sizes/ ux
- Drop-down menus (don't work)
- Group nesting (not available)
- Parameter visibility toggles (not available)

Into the Fun Stuff

Ie. Inventor Modeling

Simple Toggles

Yes… ‘Simple’… it’s up-hill from here

Talking over Video time:

Video Not Available

Mucho Feature Toggling

What is this ‘patience’ you refer to??

Talking over Video time:

Slow Feature Toggling

```
For x = 1 To Step_Count
Feature.IsActive("Step " & x & " Extrusion") = Toggle
Next
End If
```

Video Not Available

Toggle it like you stole it

Yay… iLogic Code

```
Sub Main()
'Break

'Get all features
Dim oDoc As PartDocument = ThisDoc.Document
Dim oFeatures As PartFeatures = oDoc.ComponentDefinition.Features

'Make a Selection Set
Dim oSelectSet As SelectSet = oDoc.SelectSet

'These are what are going to temporarily store all of our objects that we will then slecect and turn off
Dim oSelectOccOn As ObjectCollection = ThisServer.TransientObjects.CreateObjectCollection
Dim oSelectOccOff As ObjectCollection = ThisServer.TransientObjects.CreateObjectCollection

'A list of params that we will toggle
Dim oTurnOff = New ArrayList
Dim oTurnOn = New ArrayList
oTurnOff.Clear
oTurnOn.Clear
```

Setup a bunch of cases for when features are On or Off

```
If Arrow Then
    Select Arrow_Type
    Case 0
        oTurnOn.Add("Arrow Tip")
        oTurnOff.Add("Arrow Stem")
        oTurnOff.Add("Arrow Tip Fill")
        oTurnOff.Add("Arrow Radii")
    Case 1
        oTurnOn.Add("Arrow Tip")
        oTurnOn.Add("Arrow Stem")
        oTurnOff.Add("Arrow Tip Fill")
        oTurnOff.Add("Arrow Radii")
    Case 2
        oTurnOn.Add("Arrow Tip")
        oTurnOn.Add("Arrow Stem")
        oTurnOn.Add("Arrow Tip Fill")
        oTurnOff.Add("Arrow Radii")
    Case 3
        oTurnOff.Add("Arrow Tip")
        oTurnOff.Add("Arrow Stem")
        oTurnOff.Add("Arrow Tip Fill")
        oTurnOn.Add("Arrow Radii")
    End Select
Else
```

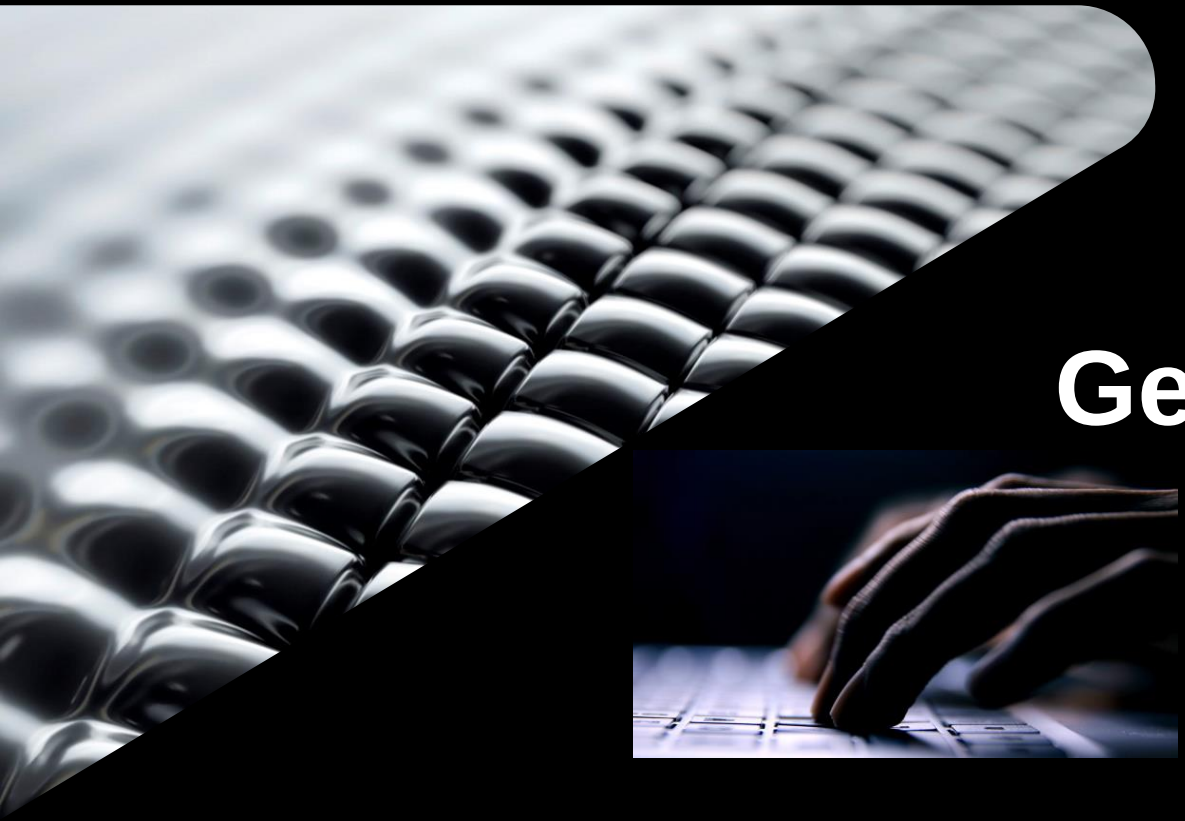
```
ElseIf FeatureName = Check & Sign_Style & " Rads" Then
    If Rads Then
        Logger.Debug(FeatureName & "++ON")
        oTurnOn.Add(FeatureName)
    Else
        Logger.Debug(FeatureName & "--Off")
        oTurnOff.Add(FeatureName)
    End If
```

```
'Get the actual objects from name and add them to a collection of object
For Each oPartFeature As PartFeature In oFeatures
    For Each name As String In oTurnOff
        If oPartFeature.Name = name Then
'           Logger.Debug("Turn-Off: " & name)
            If Not oPartFeature.Suppressed Then
            oSelectOccOff.Add(oPartFeature)
            End If
        End If
    Next
    For Each name As String In oTurnOn
        If oPartFeature.Name = name Then
'           Logger.Debug("Turn-ON: " & name)
            If oPartFeature.Suppressed Then
            oSelectOccOn.Add(oPartFeature)
            End If
        End If
    Next
Next
```

Turn off/on Features en masse

```
'Turn Off Selected Parts
oDoc.ComponentDefinition.SuppressFeatures(oSelectOccOff)

'Turn On Selected Parts
oDoc.ComponentDefinition.UnsuppressFeatures(oSelectOccOn)
InventorVb.DocumentUpdate()
End Sub
```

BONUS: How to do Select/Case



Getting into the Weeds

You’re all programmers right??

Parameters-a-palooza

Do you take the BLUE parameter, or the RE...gular parameter?

Show in iLogic that there are a couple different types of parameter call outs:

	E.g.
Parameter.Param (“{parameter name}”)	**Parameter.Param(“Step_1”)**
Parameter(“{parameter name}”)	**Parameter(“Step_1”)**

Can use variables to call out the parameters

{parameter name}	**Step_1**

Triggers the code upon parameter change

TriggeredBy = **ThisParameter** & _
NextParameter

Next Line

```
StepX_Width = Parameter(This_Step & "_Width")
StepX_Height = Parameter(This_Step & "_Height")
StepX_LHSideAngle = Parameter(This_Step & "_LHSideAngle")
StepX_RHSideAngle = Parameter(This_Step & "_RHSideAngle")
GirderX_SlopeInDeg = Parameter("Girder" & Step_Num & "_SlopeInDeg")
```

```
If AbtSlopes_Toggle = True Then
    For x = 1 To MaxGirders
    Parameter.Param("Step" & x & "_LHSideAngle").Value = Math.PI / 2
    Parameter.Param("Step" & x & "_RHSideAngle").Value = Math.PI / 2
    Next
End If
```

```
Math.Tan(Step1_LHSideAngle * inRadians)
```

BONUS:

```
inRadians = Math.PI / 180
```

This… is Important

The single-line of code that will save HOURS of beating your head against a wall

Common part of coding is to actually tell the code what it's supposed to act upon… seems pretty obvious.

`ThisApplication.`

Inventor Pro

`ThisServer.`

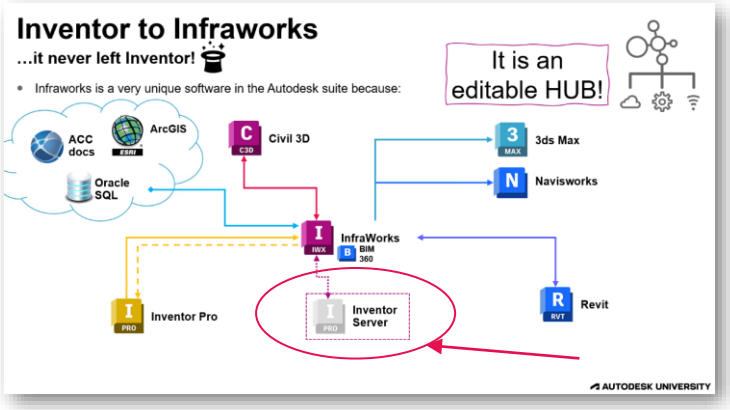
Inventor Server

Also works in regular Inventor

`ThisDoc.`

Open Document

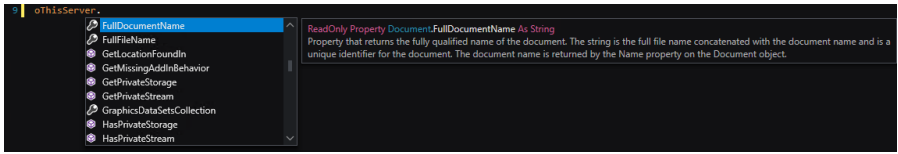
Essentially: Whatever's open in Inventor



Key sections of the code:

```
Dim oPartDoc As PartDocument = ThisDoc.Document
Dim oAssyDoc As AssemblyDocument = ThisDoc.Document
Dim oDoc As Document = ThisDoc.Document

Dim oThisApplication As Document = ThisApplication.Documents.Item(1)
Dim oThisServer As Document = ThisServer.Documents.Item(1)
Dim oDoc As Document = ThisDoc.Document
```

*Properly declaring = Getting the **helpful pop-ups** in the code*



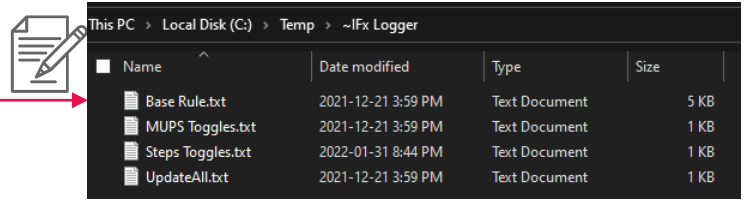
Cutting Logs

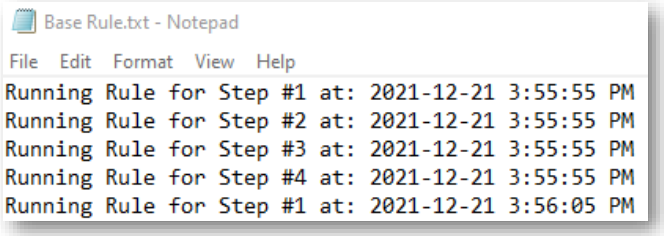
So you can finally figure out why/when Infraworks is breaking your model!

iLogic is not restricted to just Inventor…

So the Inventor server can **ALSO** create and manipulate things outside of Inventor…

…And outside of Infraworks





Key sections of the code:

```
Try
    oWrite = System.IO.File.AppendText("C:\Temp\~IFx Logger\Base Rule.txt")
Catch
    oWrite = System.IO.File.CreateText("C:\Temp\~IFx Logger\Base Rule.txt")
End Try

oWrite.WriteLine("Running Rule for Step #" & RuleArguments("StepCount") & " at: " & System.DateTime.Now)
oWrite.Close()
```

Anything you want from your code can be put to your very own log files

That Don’t Suppress Me Much

If you don’t get the reference, Google Shania Twain

In Inventor there is **Enabled**, **Visible**, **Transparent**, **Suppressed**, for **Features**, **Bodies**, **Parts**, **Assemblies**…

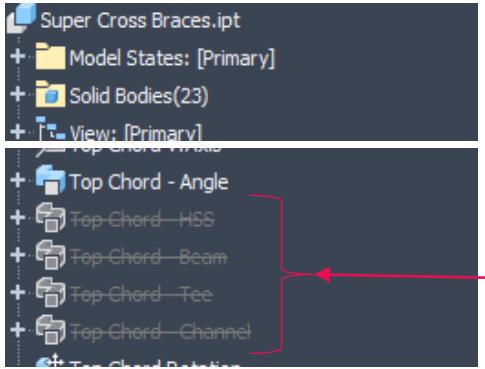
It gets confusing quickly

Easy Guide:

Parts

Suppressing a feature

Feature.IsActive(“{Feature Name}”) = **True** or **False**



Suppressed

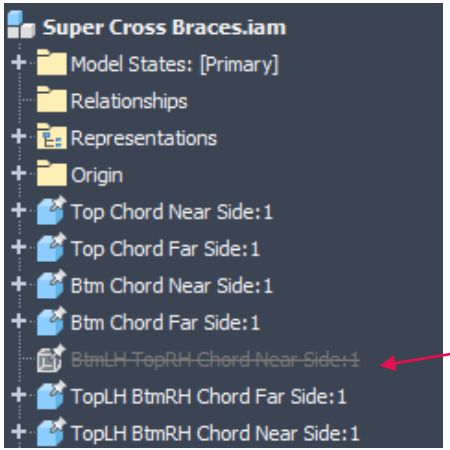
Suppressing a body

Don’t’ bother…

Assemblies

Suppressing a Part/Assy

Component.IsActive(“{Component Name}”) = **True** or **False**



Suppressed

‘Components’ refers to both PARTS and SUB-ASSEMBLIES

Changing Colours

Because… sometimes you want to change colours

```
1   Dim oPartDoc As PartDocument = ThisDoc.Document
2   Dim oPartDef As ComponentDefinition = oPartDoc.ComponentDefinition
3   Dim oFeatures As PartFeatures = oPartDef.Features
4   Dim libAsset1 As Asset
5
6   Dim BorderColor_Asset As Asset = oPartDoc.AppearanceAssets.Item(Border_Color)
7   Dim TextColor_Asset As Asset = oPartDoc.AppearanceAssets.Item(Text_Color)
8   Dim ArrowColor_Asset As Asset = oPartDoc.AppearanceAssets.Item(Arrow_Color)
9
10  libAsset1 = oPartDoc.AppearanceAssets.Item(Sign_Color)
11  oPartDoc.ComponentDefinition.SurfaceBodies.Item(1).Appearance = libAsset1
12
13  'Get the actual objects from name and add them to a collection of object
14  For Each oPartFeature As PartFeature In oFeatures
15          If Right(oPartFeature.Name, Len("Border")) = "Border" Then
16              oPartFeature.Appearance = BorderColor_Asset
17              Logger.Debug("Border Found")
18          ElseIf Left(oPartFeature.Name, Len("Text")) = "Text" Then
19              oPartFeature.Appearance = TextColor_Asset
20              Logger.Debug("Text Found")
21          ElseIf Left(oPartFeature.Name, Len("Arrow")) = "Arrow" Then
22              oPartFeature.Appearance = ArrowColor_Asset
23              Logger.Debug("Arrow Found")
24          End If
25  Next
```

General Declarations

Declarations, setting assets to match parameter values

In this example, parameter names exactly match the asset name (e.g. Paint – Blue)

Change part **body** to color asset

Cycle through all the features…

Conditional statements for what feature(s) should be changed

Change part **feature** to color asset

Changing Text

E.g. Custom Graffiti on Piers

iLogic

Key Parts of the Code:

```
1   Dim oPartDoc As PartDocument = ThisDoc.Document
2   Dim oPartDef As ComponentDefinition = oPartDoc.ComponentDefinition
3   Dim oFontSize As Long = Text_Line1Height/10
4   Dim Check As String
5   Dim x As Integer
```

General Declarations

Convert to **mm**; default iLogic is ALWAYS **cm**

```
For Each oSketch In oPartDef.Sketches
    Check = Left(oSketch.Name,Len("Text - "))
    Logger.Debug("TEST" & x & ": " & Check)
'   x = x + 1
    If Check = "Text - " Then
        Logger.Debug("Text Sketch Found")
        x = 0
        For Each oTextbox In oSketch.TextBoxes
            x = x + 1
            Select x
            Case 1
                oTextbox.FormattedText = "<StyleOverride FontSize = '" & Text_Line1Height/10 & "'>" & Text_FirstLine & "</StyleOverride>"
            Case 2
                oTextbox.FormattedText = "<StyleOverride FontSize = '" & Text_Line2Height/10 & "'>" & Text_SecondLine & "</StyleOverride>"
            Case 3
                oTextbox.FormattedText = "<StyleOverride FontSize = '" & Text_Line3Height/10 & "'>" & Text_ThirdLine & "</StyleOverride>"
            End Select
        Next
    End If

Next

InventorVb.DocumentUpdate()
```

In other terms:

oPartDef.Sketches.Sketch({number}).TextBoxes.Textbox({number}).FormattedText

Equals:

“<StylerOverride FontSize = ‘{font size value}’> {your text} </StyleOverride>”

There’s more you can change, just google the style override commands.

WTF?!

That’s right… it’s a different code language within the iLogic code

Diving Even Deeper

Let’s just directly edit background files

Drop it like it's Hot

Likely outdated information

Notepad or Better

How to make drop-down lists work… until Autodesk finally gets around to doing it right

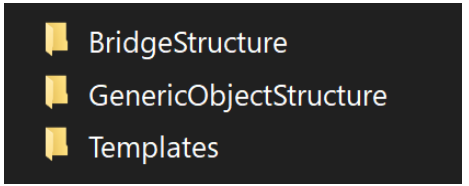
Did you know you can make drop-down menus in Infraworks?

…yah, all you have to do is go and adjust the background code of the program

IN .AcItem or .JSON files:

```
"{parameter name}": {
"Details": [ "{parameter name}", "", false, false, 34 ],
"DataType": 3,
"Enumeration": {
"Labels": [ "{label0}", "{label1}", "{label2}", "{label3}", "{label4}" ],
"Keys": [ "0", "1", "2", "3", "4" ]
}
},
```

This is what's returned to Inventor, not the label value *sigh*



Note:
Some of the bridge structure models will be in GenericObjectStructure

Path: C:\Users\{username}\Documents\Autodesk InfraWorks Models\Autodesk 360\Sandbox\1252178\{Ifx Model Name}.files\unver\Content\Parts

Generated number for the Infraworks model

Infraworks version or Sandbox

Drop-down Example

Good luck…

Likely outdated information

Notepad or Better

Note: Can also edit in JSON files

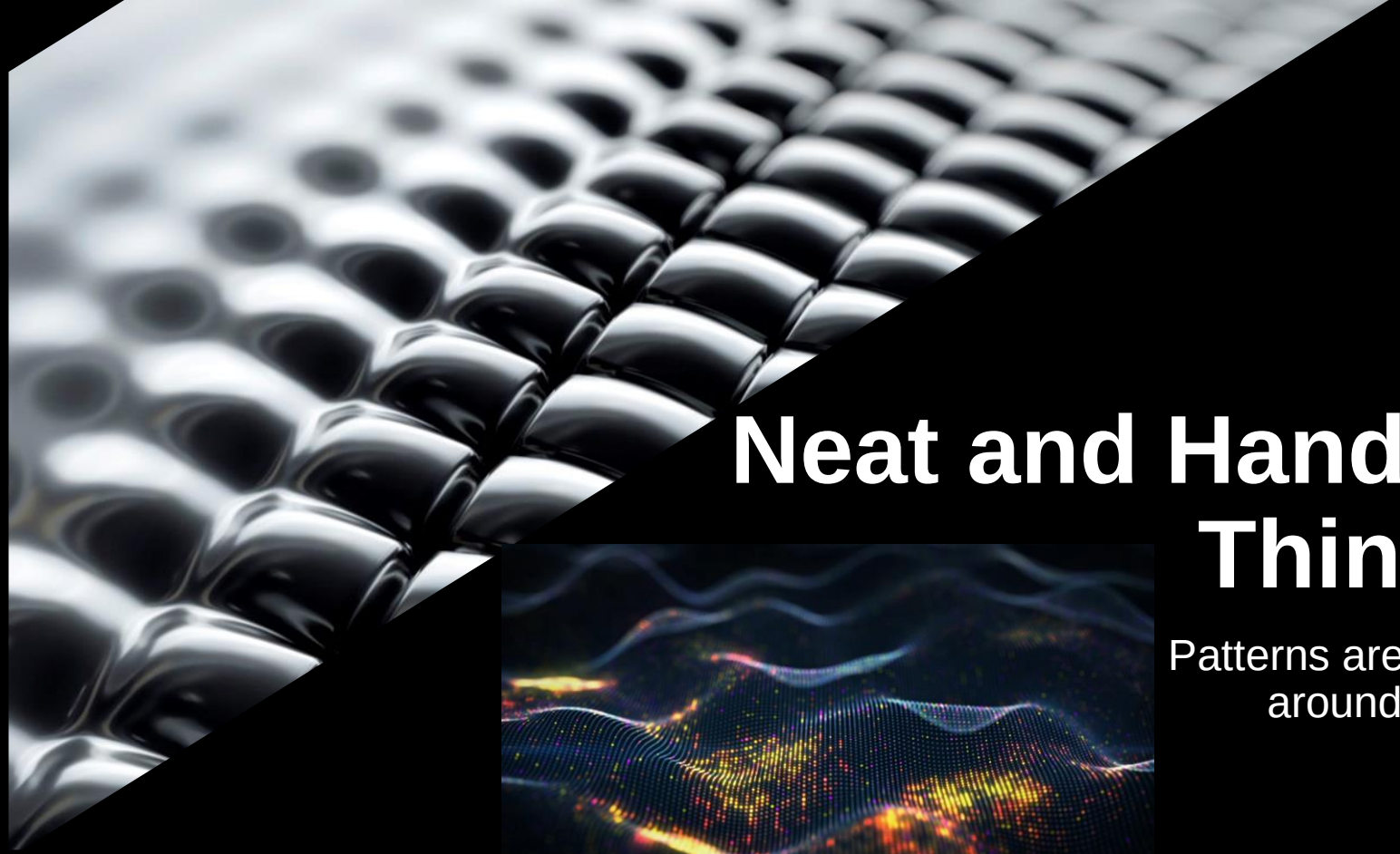
```
},
"Start_Step_Toggle": {
    "Details": [ "Step Toggle", "Toggle to include the stepped cut-out from the start of the girder", false, false, 22 ],
    "DataType": 2
},
"Start_VerticalSlope_OR": {
    "Details": [ "Vertica Slope Override", "Override (in % slope) of the vertical face value", false, false, 25 ]
},
"SuperT_BeamType": {
    "Details": [ "SuperT Beam Type", "Menu list of available SuperT beam types", false, false, 0 ],
    "DataType": 3,
    "Enumeration": {
        "Labels": [ "T1", "T2", "T3", "T4", "T5" ],
        "Keys": [ "0", "1", "2", "3", "4" ]
    }
},
"SuperT_EndLeft_dWidth": {
    "Details": [ "Left Flange delta-Width", "Dimension change to Left Flange Width of the SuperT (looking towards the midd
    "Units": "mm"
},
"SuperT_EndRight_dWidth": {
    "Details": [ "Right Flange delta-Width", "Dimension change to Right Flange Width of the SuperT (looking towards the mi
    "Units": "mm"
},
```

Name	Date modified	Type
8a0d1f4c-00ed-4e34-bf43-3f7aa27c0b64.ACItem	2022-01-27 10:40 PM	ACITEM File
8e9ab36f-772a-4ef1-a48b-0e69a58ffb0d.ACItem	2021-12-08 11:31 PM	ACITEM File
24eacbe6-f36e-45e1-9576-de48ed5fd384.ACItem	2022-01-27 10:40 PM	ACITEM File
29c4225e-a374-45bb-a576-688fd32734ae.ACItem	2022-01-27 10:40 PM	ACITEM File
29e033f2-fb7a-476d-a045-7ae6bff051bb.ACItem	2022-01-27 10:40 PM	ACITEM File
73f29f00-f8de-455a-a81b-5d6f0330f9c8.ACItem	2021-12-08 9:07 PM	ACITEM File
247f205e-996f-4512-857f-4fbb51e3ea4c.ACItem	2021-12-07 4:43 PM	ACITEM File
443c1f16-b96f-4285-8d6d-7b5ef39e5ca4.ACItem	2022-01-11 11:00 PM	ACITEM File
616ccc34-1d9a-450d-b142-e713dc2d5982.ACItem	2022-01-11 11:00 PM	ACITEM File
7337b644-f864-4630-be49-6aa9591600d2.ACItem	2022-01-27 10:40 PM	ACITEM File
42722c40-a50b-41af-8be5-89e318ac4ab7.ACItem	2021-12-14 10:54 PM	ACITEM File
738358f2-2fad-4954-af06-5402e9a2d7c7.ACItem	2022-01-27 10:40 PM	ACITEM File
a4d838f9-5b13-464f-8b22-c871f32e33bd.ACItem	2021-12-12 8:01 PM	ACITEM File
a9e3f390-1077-4ce5-a86e-a3aee62b99e4.ACItem	2021-12-07 4:43 PM	ACITEM File
a2356b62-6991-4389-8f24-ec0ec78d44a6.ACItem	2021-12-07 4:43 PM	ACITEM File
bf81d4c0-3bc4-4980-b384-89157de4a17c.ACItem	2021-12-10 11:33 PM	ACITEM File
db2172d3-da96-4b2b-b9f4-1ec1d94c3434.ACItem	2022-01-30 7:54 PM	ACITEM File
f640e734-6929-44ea-ab15-0e5d6a8a4494.ACItem	2021-12-10 11:33 PM	ACITEM File
faee3cbd-4bf3-4ed4-b061-06d2fc245cae.ACItem	2022-04-21 9:25 PM	ACITEM File

```
    },
    "PartRecords": [
        {
            "PartSizeId": "a3c88c3c-8c3b-4b99-a442-5829a1ed5a53",
            "Size": "Jan 11-2",
            "PartSizeLocalName": "Jan 11-2",
            "GeometryParamValues": {
                "Closed_Flange": 0,
                "Depth1": 1800,
                "Depth2": 1800,
                "Diaphragm_CLOffset": 0,
                "Diaphragm_ChamferWidth": 150,
                "Diaphragm_Count": 3,
                "Diaphragm_Spacing": 4000,
```

```
"Start_StepDistance": 1350,
"Start_Step_Toggle": 1,
"Start_VerticalSlope_OR": 0,
"SuperT_BeamType": "T5",
"SuperT_EndLeft_dWidth": 0,
"SuperT_EndRight_dWidth": 0,
"SuperT_LeftMiddleOR": 0,
```

Your chosen default value



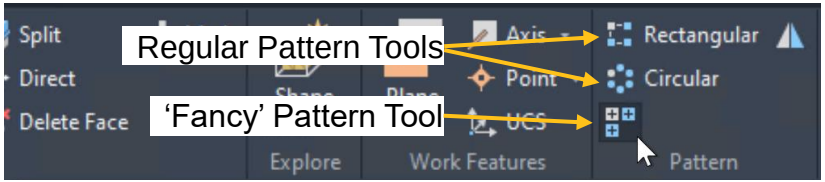
Neat and Handy Thing

Patterns are all around us

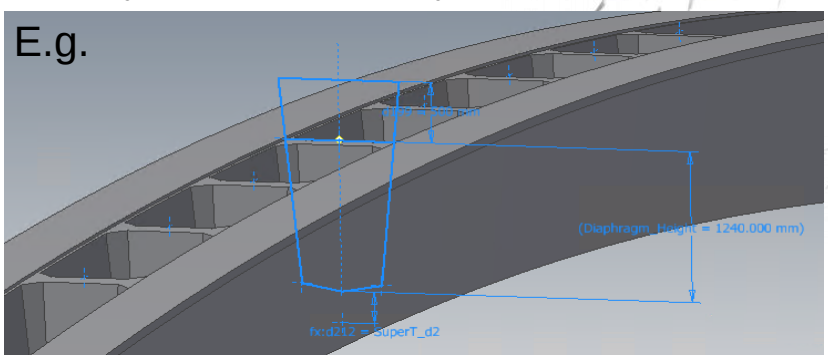
By Patternatizing

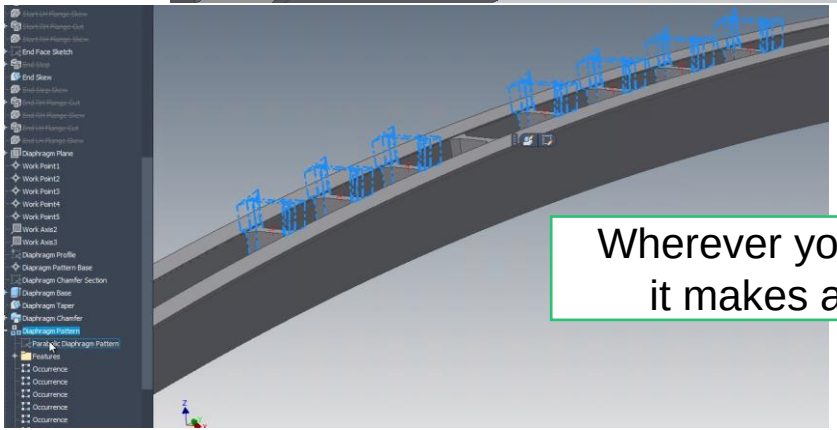
….{pithy comment here}…

Sometimes you want pattern things… but in a really messed-up way…



E.g.





Wherever you put a Point, it makes an instance

Connecting the Dots

I <u>DARE YOU</u> to find another way to do what this does…

Key sections of the code:

```
Dim oDoc As PartDocument = ThisDoc.Document
Dim oDef As PartComponentDefinition = oDoc.ComponentDefinition
Dim oTG As TransientGeometry = ThisServer.TransientGeometry
Dim oTO As TransientObjects = ThisServer.TransientObjects

Dim oPoint As SketchPoint
Dim NewPoint As Point2d

Dim NewPointY As Double
Dim i As Integer
Dim Interval As Integer
Dim Flip As Integer = 1

Dim PatternSketch As Sketch = oDef.Sketches.Item("Parabolic Diaphragm Pattern")
Dim PointCount As Integer

PointCount = PatternSketch.SketchPoints.Count
Logger.Info("Starting PointCount == " & PointCount)

Try
    For Each oPoint In PatternSketch.SketchPoints
        oPoint.Delete
    Next
Catch
End Try
```

Define Transients

Define Points

Point2d needs X & Y both defined

Delete all the old points

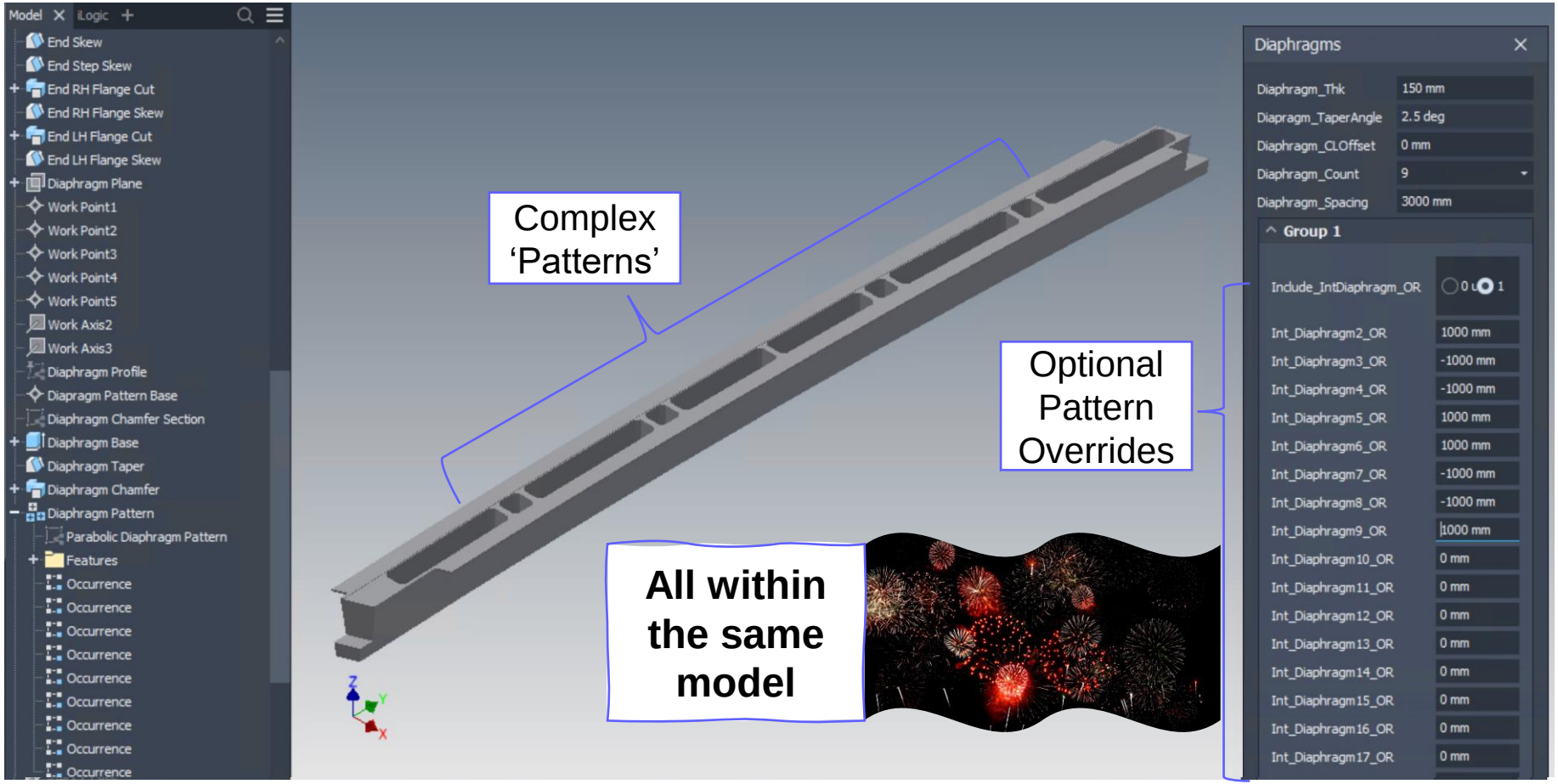
```
NewPointY = (Interval * Diaphragm_Spacing) * Flip + Diaphragm_CLOffset + Diaphragm_OR
Logger.Debug("NewPointY =" & Interval & "= " & NewPointY)
NewPoint = AddPoint(NewPointY)
If Not NewPoint.X = 50000 Then
PatternSketch.SketchPoints.Add(NewPoint)
End If
Flip  = -Flip
```

Add new points

```
Public Function AddPoint(Y_Coord As Double)

Dim oTG As TransientGeometry = ThisServer.TransientGeometry
Dim Coord(1) As Double
Dim Add_Point As Point2d

If Y_Coord > CurveEqn_StartDiaphragmX And Y_Coord < CurveEqn_EndDiaphragmX Then

Coord(0) = Y_Coord*(1/10)
Coord(1) = (-(Y_Coord^2)/((Half_Segment^2)/Hog))*(1/10)

Logger.Info("New Point Added at: " & Coord(0)*10 & "|Y & " & Coord(1)*10 & "|Z")

AddPoint = oTG.CreatePoint2d(Coord(0),Coord(1))

UsedDiaphragms = UsedDiaphragms + 1
    Logger.Debug("UsedDiaphragms =" & UsedDiaphragms)
Return AddPoint

Else
AddPoint = oTG.CreatePoint2d(50000,0)
Return AddPoint

End If
```

Function for how you want to add your points

Then you can go from this… To this…



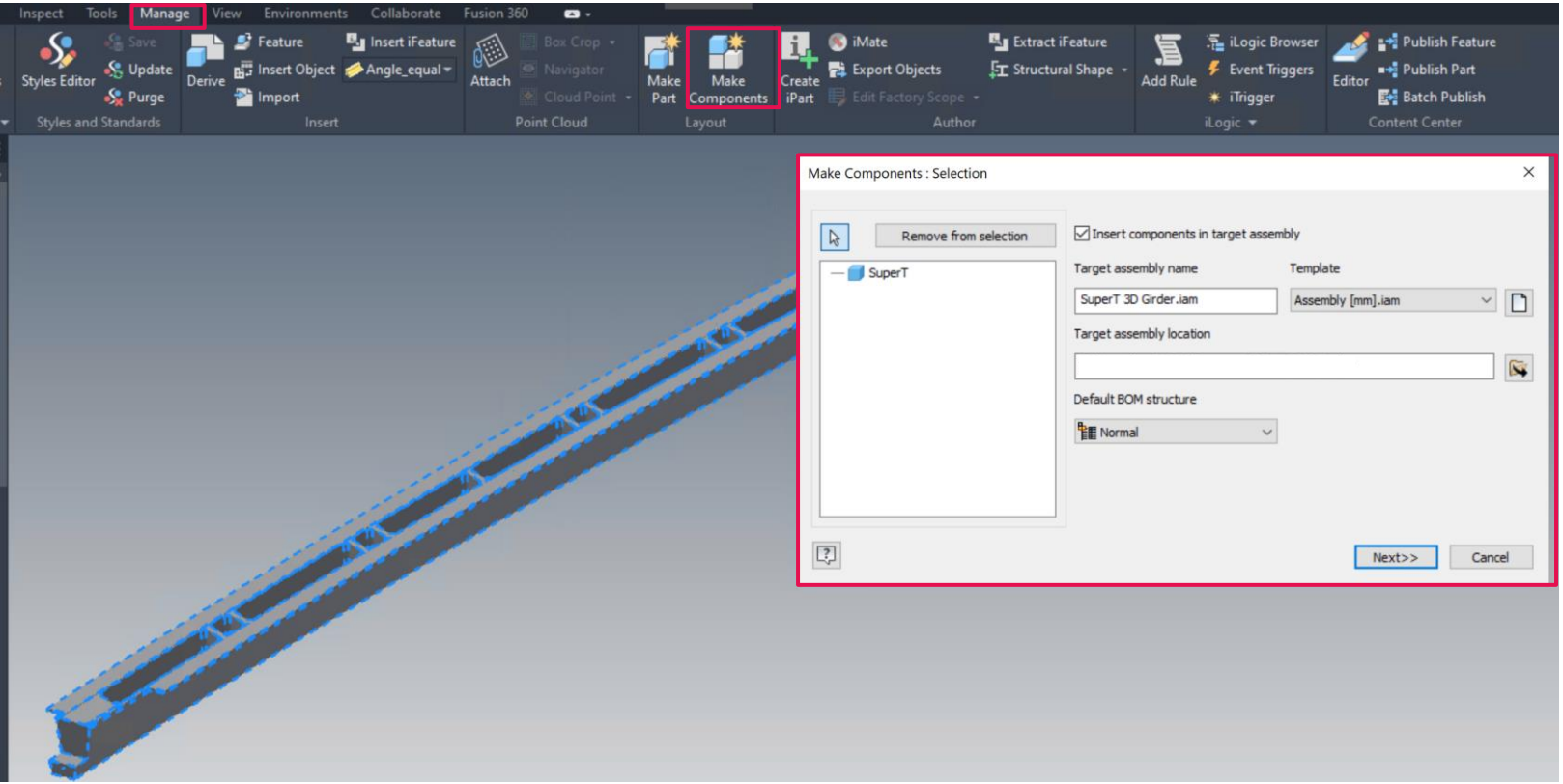
Victory Lap

Presentation is done…

Now I’m just showing off

Multi-Body Parts to Assembly

Blatantly just showing off our internally developed tools… I guess you should just hire Hatch :P



Except it sucks!

So we built our own tools…

Video Not Available