ETO Potlatch

Comparison of Positioning Techniques

Vol. 1, 6-Sep-2012

Online Resources:

Forum:	http://forums.autodesk.com/t5/Autodesk-Inventor-Engineer-to/bd-p/184
Wiki help:	http://wikihelp.autodesk.com/Inventor_ETO/enu/2013

Autodesk

Welcome / Agenda

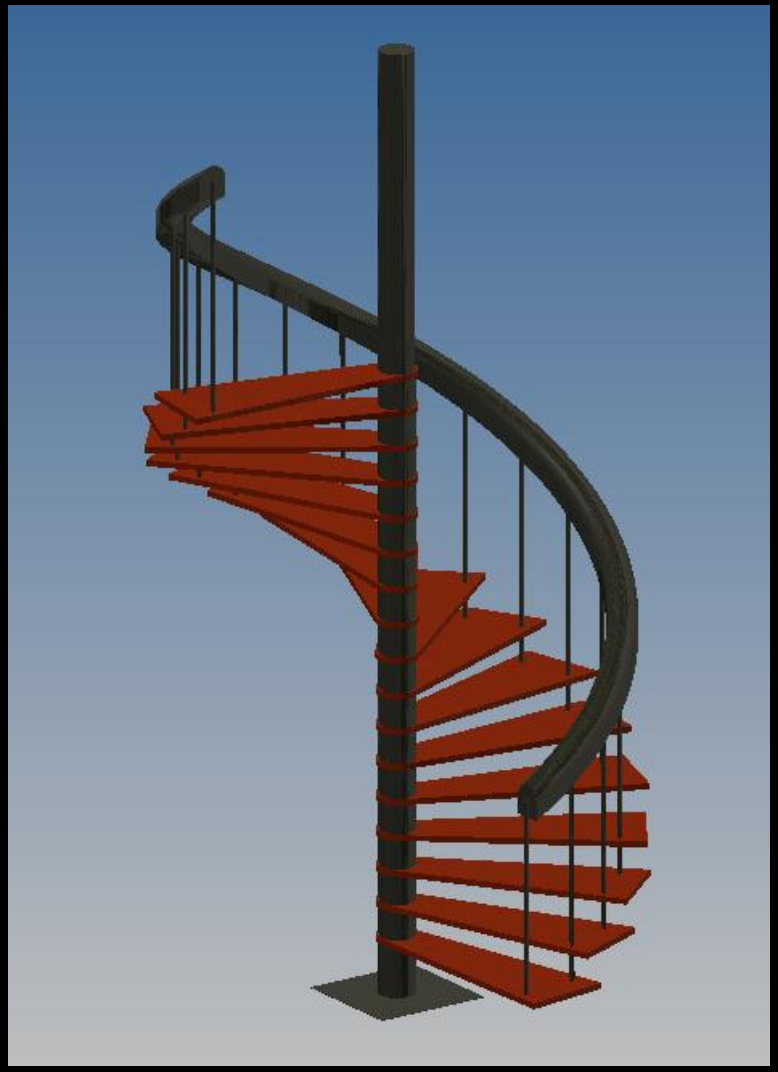
- 30 minutes presentation on selected topic:
  - “Comparison of Positioning Techniques” – Jon Balgley
- 20 minutes Q&A and discussion “on-topic”
- 10 minutes “Three Tips”
- 20 minutes Q&A and discussion “on any topic”

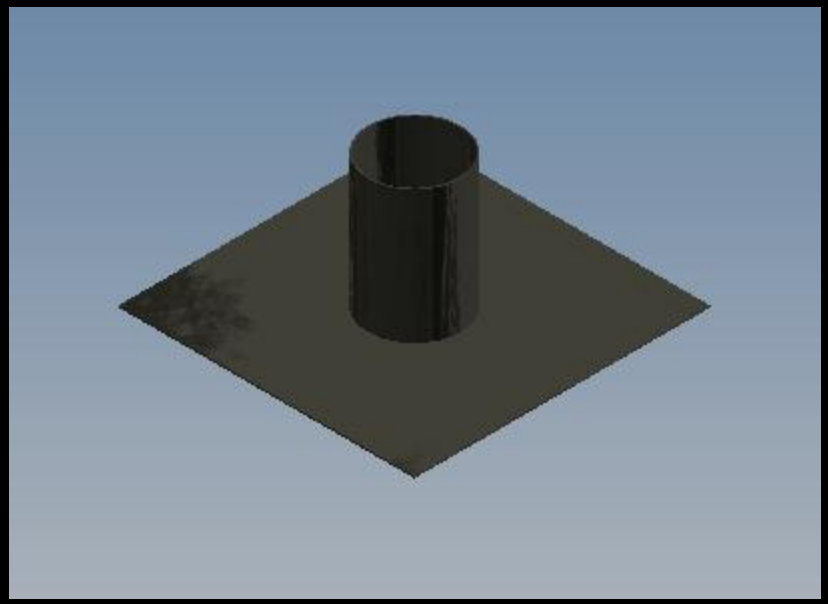
Positioning Techniques

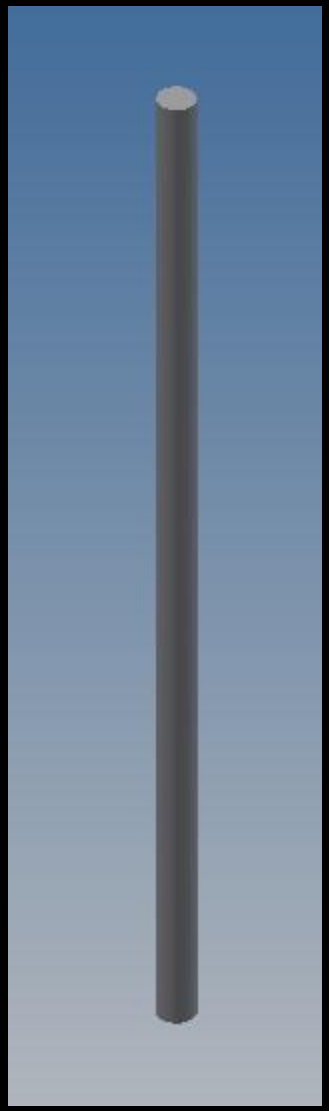
- The two options
- How they work
- Analysis/comparison/timings

Positioning Techniques

- Constraint-based Positioning (CBP)
- Frame-based Positioning (FBP)







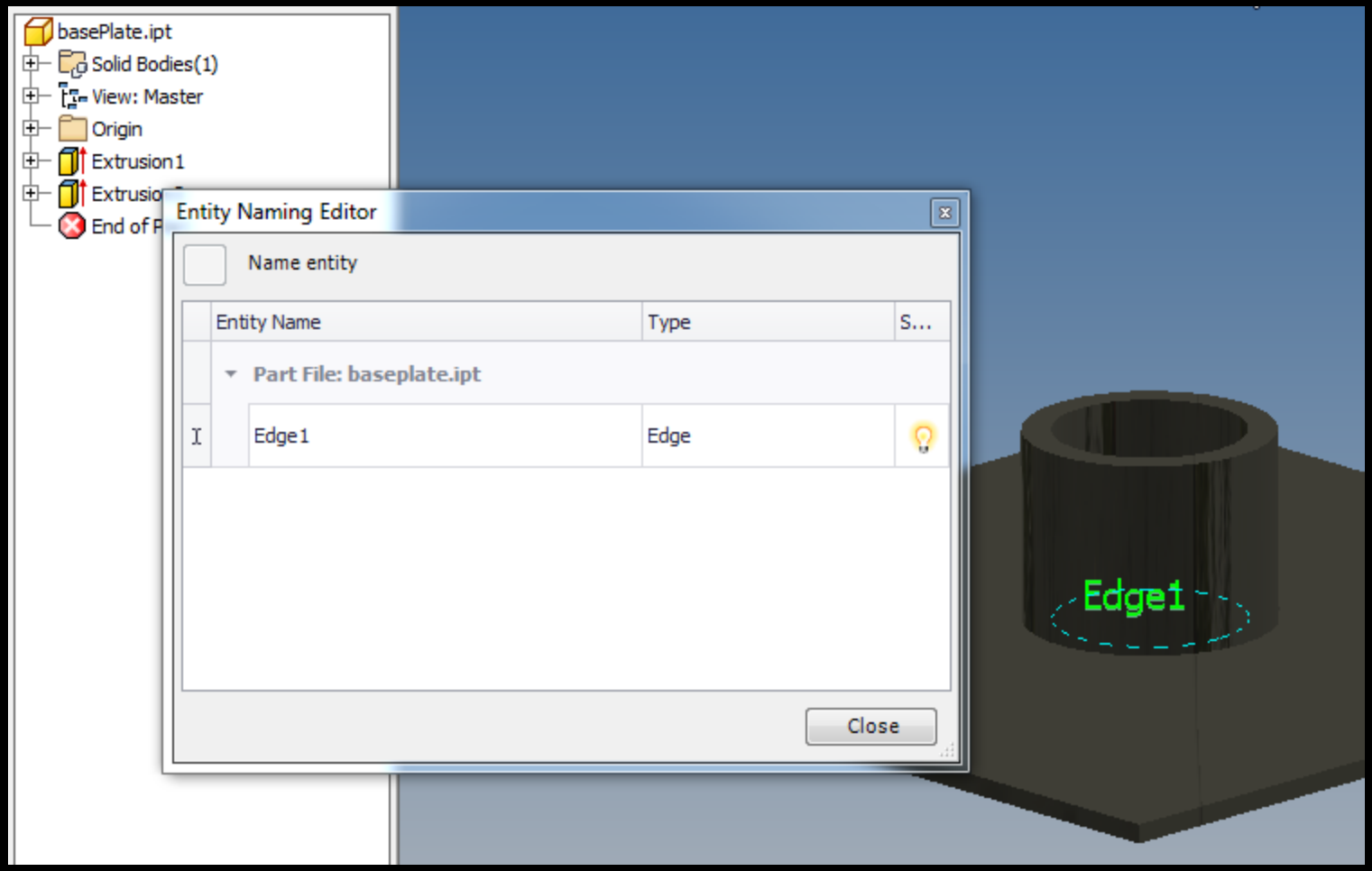
Autodesk

CBP

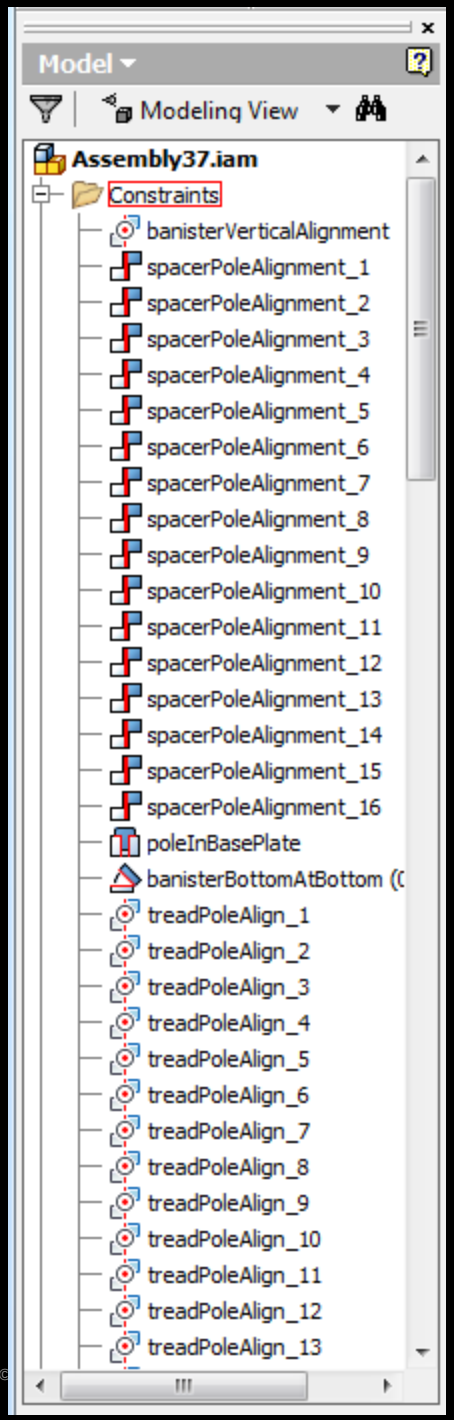
```
Child pole As :IvCylinder
    height = poleHeight
    diameter = poleDiameter
    color = poleMaterial
End Child

Child basePlate As :basePlate
    baseSize = 500
    baseThickness = 3
    collarHeight = firstTreadHeight - tread_1.tread.thickness
    collarThickness = 3
    poleDiameter = poleDiameter
    grounded? = True
    color = spacerMaterial
End Child
```

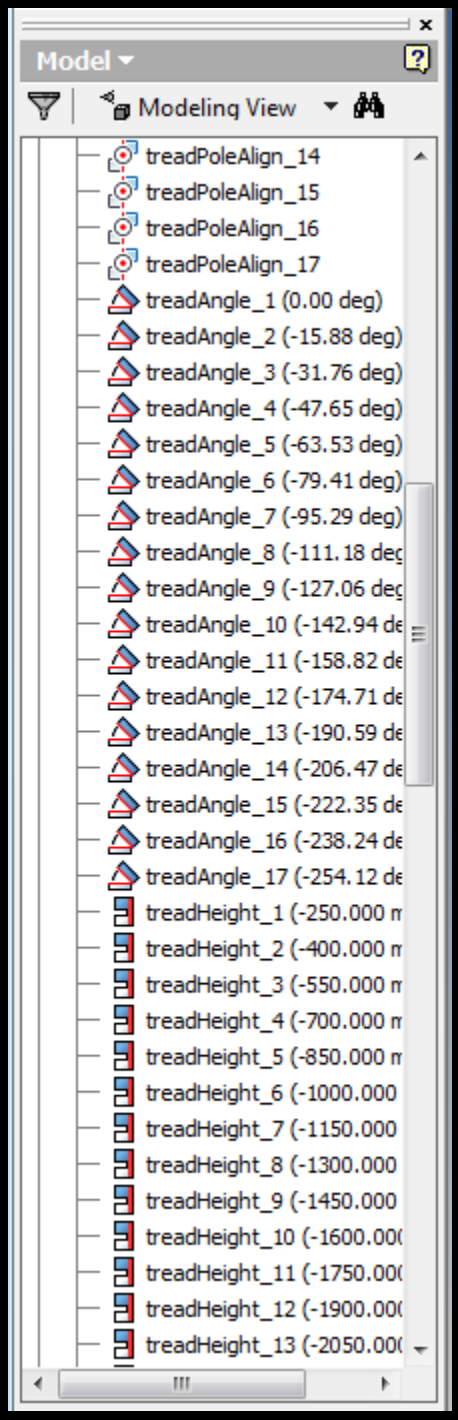
```
Child poleInBasePlate As :IvInsertConstraint
    part1 = pole
    entity1 = "eBottom"
    part2 = basePlate
    entity2 = "Edge1"
    distance = 0
    axesOpposed? = True
End Child
```

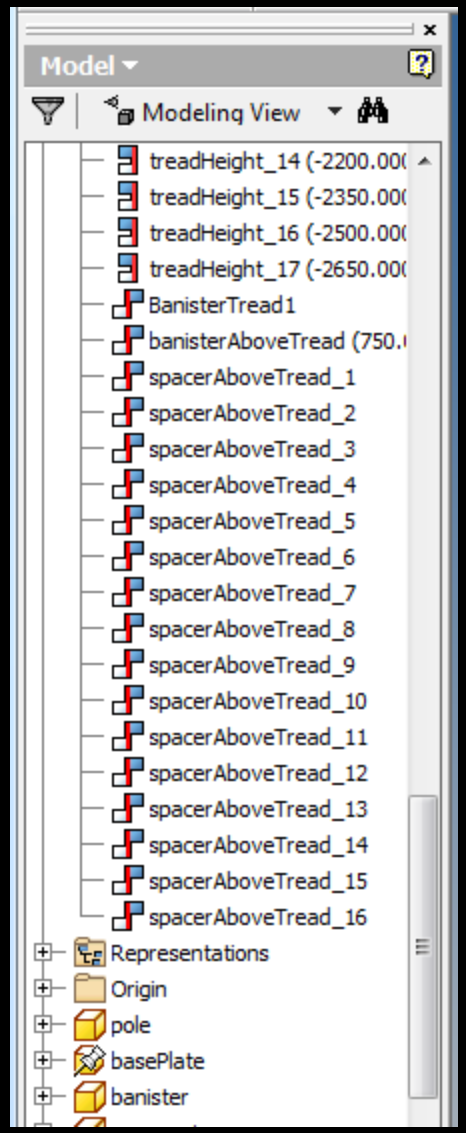
- Only correct relative to each other
- Where do rules place the ‘basePlate’?
- Is this code error-free?



Autodesk


Model
Modeling View
Assembly37.iam
Constraints
banisterVerticalAlignment
spacerPoleAlignment_1
spacerPoleAlignment_2
spacerPoleAlignment_3
spacerPoleAlignment_4
spacerPoleAlignment_5
spacerPoleAlignment_6
spacerPoleAlignment_7
spacerPoleAlignment_8
spacerPoleAlignment_9
spacerPoleAlignment_10
spacerPoleAlignment_11
spacerPoleAlignment_12
spacerPoleAlignment_13
spacerPoleAlignment_14
spacerPoleAlignment_15
spacerPoleAlignment_16
poleInBasePlate
banisterBottomAtBottom (
treadPoleAlign_1
treadPoleAlign_2
treadPoleAlign_3
treadPoleAlign_4
treadPoleAlign_5
treadPoleAlign_6
treadPoleAlign_7
treadPoleAlign_8
treadPoleAlign_9
treadPoleAlign_10
treadPoleAlign_11
treadPoleAlign_12
treadPoleAlign_13

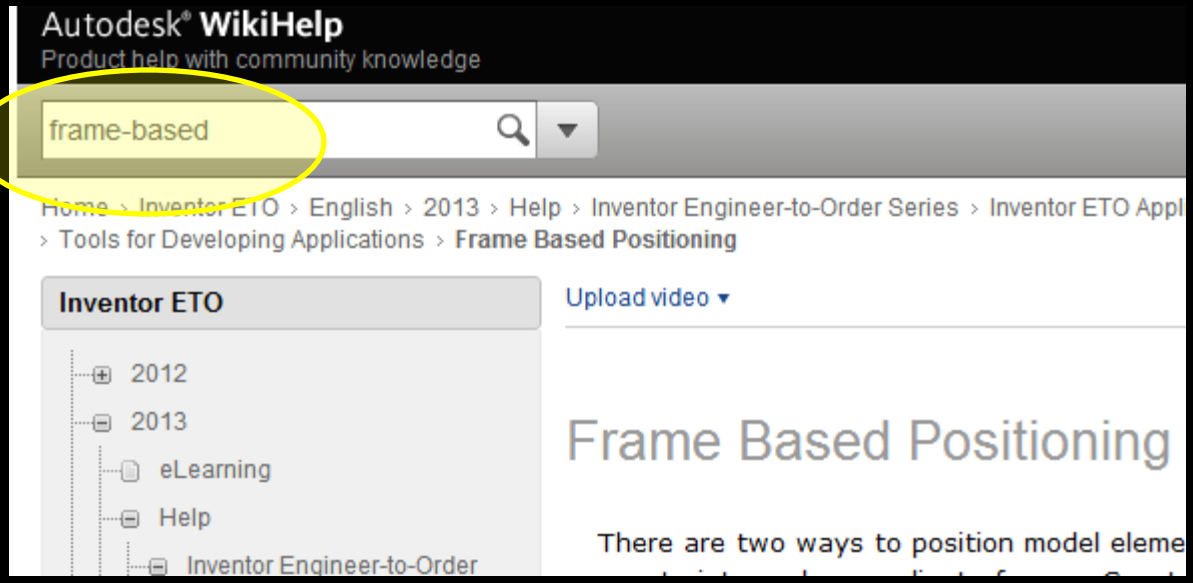

Model
Modeling View
treadPoleAlign_14
treadPoleAlign_15
treadPoleAlign_16
treadPoleAlign_17
treadAngle_1 (0.00 deg)
treadAngle_2 (-15.88 deg)
treadAngle_3 (-31.76 deg)
treadAngle_4 (-47.65 deg)
treadAngle_5 (-63.53 deg)
treadAngle_6 (-79.41 deg)
treadAngle_7 (-95.29 deg)
treadAngle_8 (-111.18 deg
treadAngle_9 (-127.06 deg
treadAngle_10 (-142.94 de
treadAngle_11 (-158.82 de
treadAngle_12 (-174.71 de
treadAngle_13 (-190.59 de
treadAngle_14 (-206.47 de
treadAngle_15 (-222.35 de
treadAngle_16 (-238.24 de
treadAngle_17 (-254.12 de
treadHeight_1 (-250.000 m
treadHeight_2 (-400.000 m
treadHeight_3 (-550.000 m
treadHeight_4 (-700.000 m
treadHeight_5 (-850.000 m
treadHeight_6 (-1000.000
treadHeight_7 (-1150.000
treadHeight_8 (-1300.000
treadHeight_9 (-1450.000
treadHeight_10 (-1600.00(
treadHeight_11 (-1750.00(
treadHeight_12 (-1900.00(
treadHeight_13 (-2050.00(


Model
Modeling View
treadHeight_14 (-2200.00(
treadHeight_15 (-2350.00(
treadHeight_16 (-2500.00(
treadHeight_17 (-2650.00(
BanisterTread1
banisterAboveTread (750.(
spacerAboveTread_1
spacerAboveTread_2
spacerAboveTread_3
spacerAboveTread_4
spacerAboveTread_5
spacerAboveTread_6
spacerAboveTread_7
spacerAboveTread_8
spacerAboveTread_9
spacerAboveTread_10
spacerAboveTread_11
spacerAboveTread_12
spacerAboveTread_13
spacerAboveTread_14
spacerAboveTread_15
spacerAboveTread_16
Representations
Origin
pole
basePlate
banister

Other CBP technique

- Incremental adopt

FBP



Autodesk

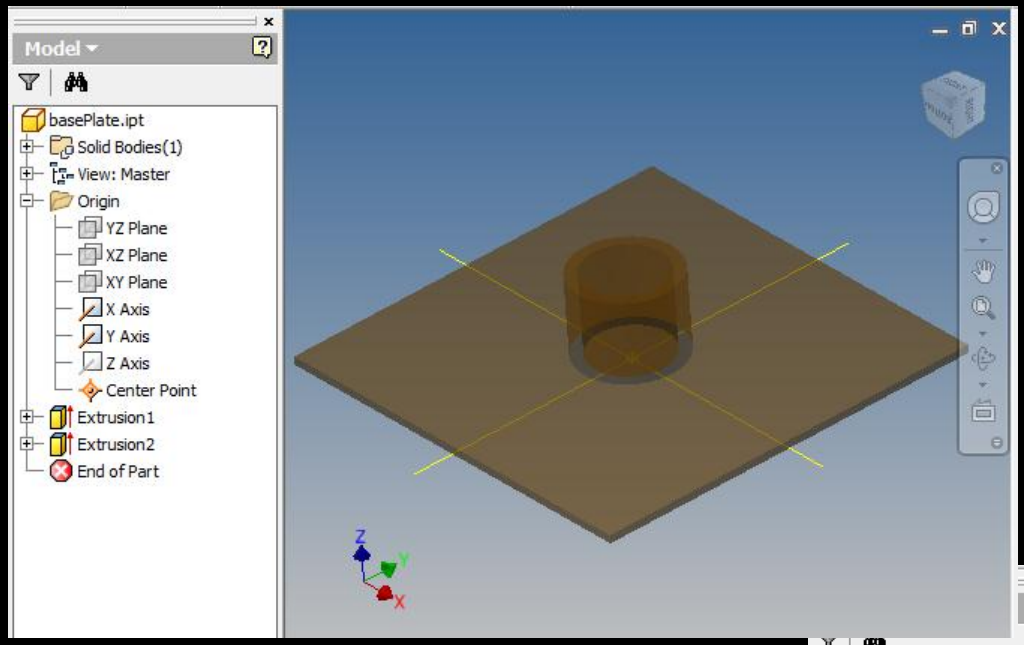
What is a Frame?

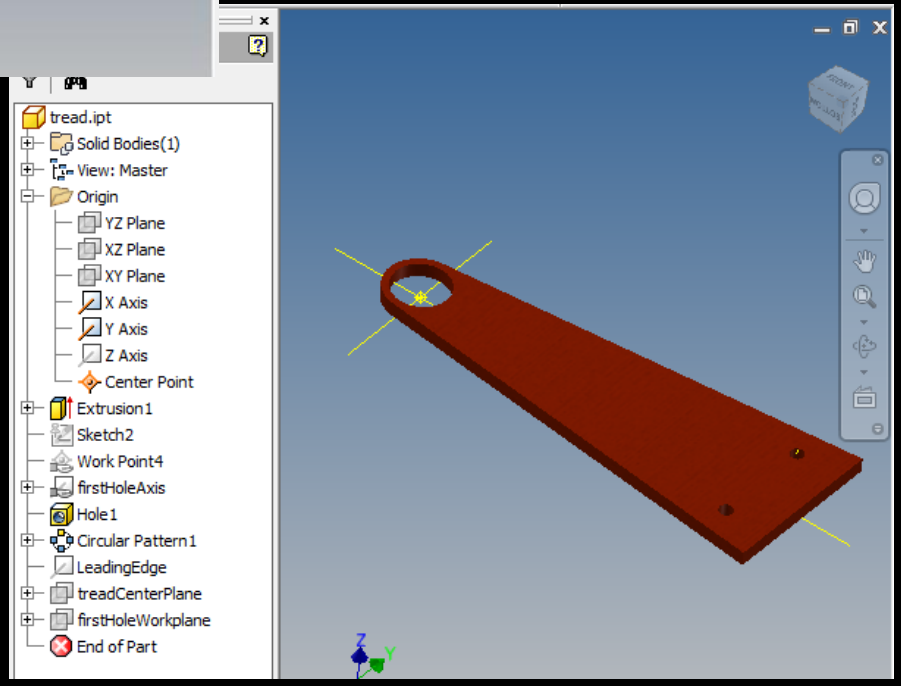
- Representation of component position & orientation
  - (a.k.a. “transform matrix”)

- Intent data type
  - never needed when using CBP
  - Often computed/used indirectly from points & vectors

- Frames are used to align “native” Origin, xDirection, yDirection to another position/orientation

- The “native” directions:
  - → Aligned with Inventor “Origin” work-elements
  - → Not *necessarily* aligned with any geometry



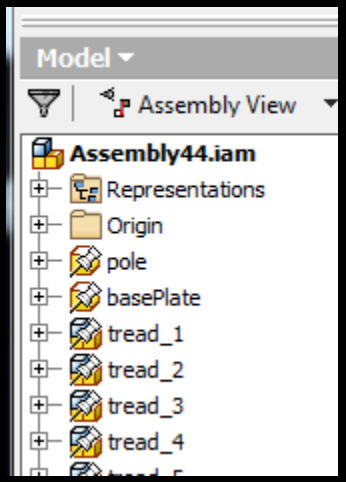


Autodesk

FBP

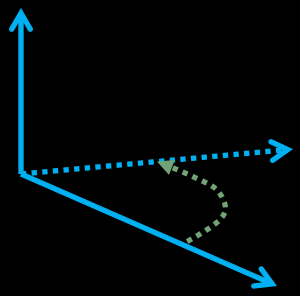
```
Child pole As :IvCylinder
    height = poleHeight
    diameter = poleDiameter
    color = poleMaterial
    bottomPoint = origin
    ignorePosition? = False
    grounded? = True
End Child
```

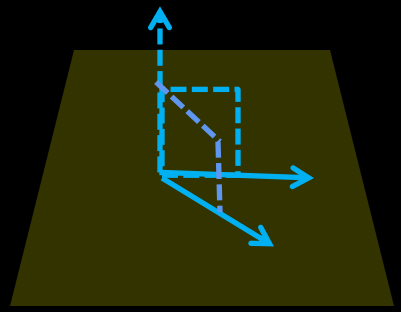
```
Child basePlate As :basePlate
    baseSize = 500
    baseThickness = 3
    collarHeight = firstTreadHeight - tread_1.tread.thickness
    collarThickness = 3
    poleDiameter = poleDiameter
    color = spacerMaterial
    origin = origin
    grounded? = True
    ignorePosition? = False
End Child
```



Autodesk

```
Child tread As :treadAssy, Quantity = nTreads
    poleDiameter = pole.diameter
    banisterRadius = banister.radius
    treadWidth = treadWidth
    stairAngle = anglePerTread
    nHoles = nBalustersPerTread
    treadRise = (If child.first? Then firstTreadHeight Else treadRise)
    firstTread? = Child.first?
    lastTread? = Child.last?
    origin = origin + Vector(0, 0, (((child.index-1) * treadRise) + firstTreadHeight))
    xDirection = RotateVector(unitX, (child.index - 1) * anglePerTread, unitZ)
    yDirection = unitZ * Child.xDirection
    grounded? = True
    ignorePosition? = False
End Child
```

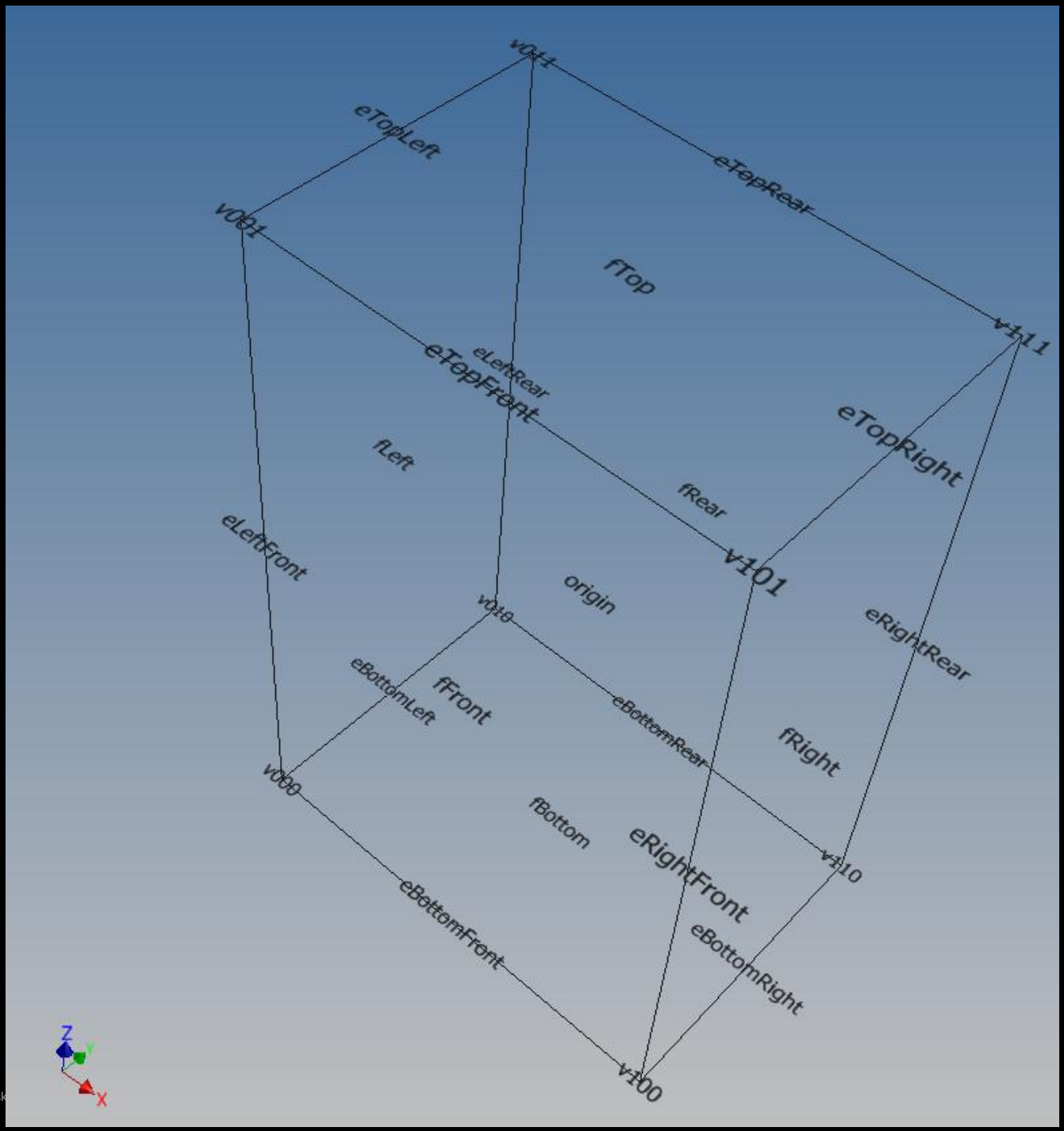




Autodesk

Other FBP Techniques

- Many point/vector/frame functions and operations
- Adopt (in 6.0) captures initial component position
- “BlockMixin” – define component L/W/H and get many useful named vertices



Autodesk

CBP Analysis

Advantages:

- Resulting Inventor assembly files are constrained
- Easier(?) to get started (can use “adopt”)
- Inventor users already understand constraints
- Easier to use, with geometrically complex parts/assemblies

Disadvantages:

- Does not work with non-Inventor ETO (e.g., web server)
- Positioning is unpredictable when constraints have multiple solutions, or under-constrained
- Harder to debug (can’t tell why constraint is “sick”)
- Sometimes requires “fully constrained” scenario to be robust
- Generally slower than FBP, sometimes much slower

FBP Analysis

Advantages:

- Works with both Inventor-based ETO and non-Inventor ETO (e.g., web server)
- Generally faster than CBP, sometimes much faster
- Positioning is always “fully constrained”, no ambiguity due to multiple solutions or under-constraining

Disadvantages:

- Resulting Inventor file has no constraints!
- Harder(?) to get started in Inventor
  - (e.g., “adopt” gets absolute position, not a useful rule)
- Perhaps harder to learn? If you have never done it before, or are afraid of a “vector”

Don’t Mix Modes!

- Constraints don’t respect FBP positioning
- FBP frames aren’t affected by constraints
- Theoretically possible to make it work, if you’re REALLY careful
  - … at different assembly levels

Why choose one or the other?

- CBP is your *only* choice if:
  - You need “well-constrained” files, to do downstream editing
- FBP is your *only* choice if:
  - You need to run your rule-set in a non-Inventor ETO (web server) environment

But which is better?

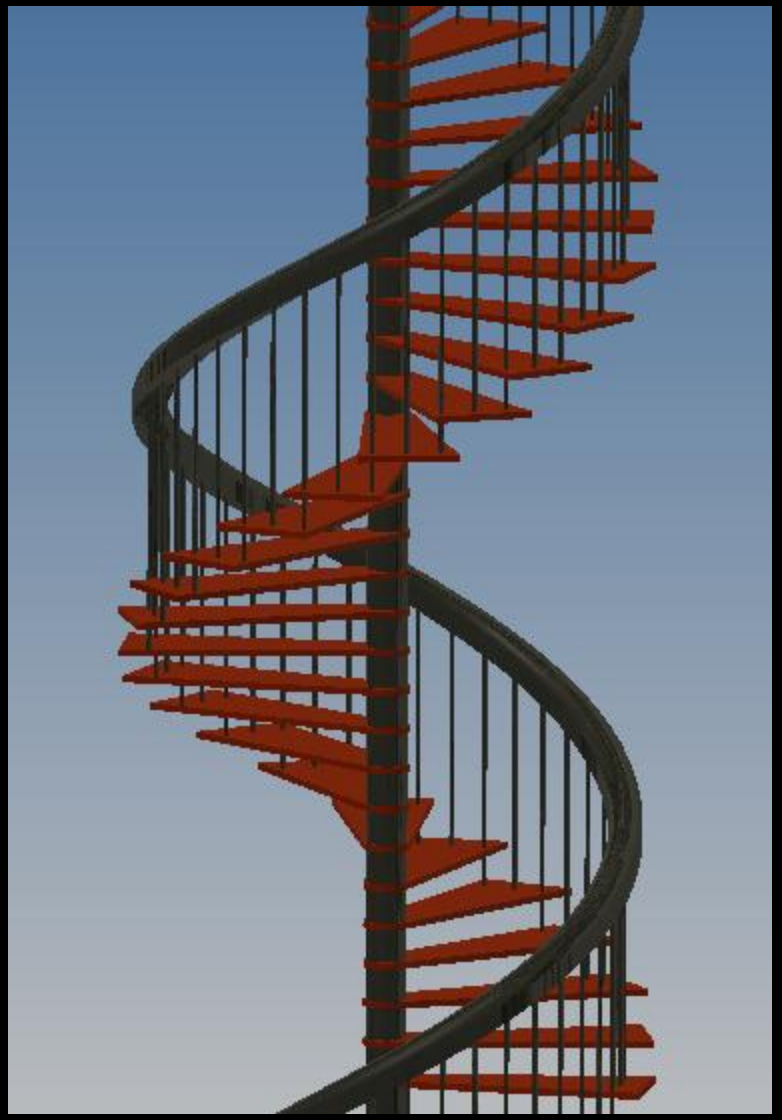
- CBP is more Inventor-ish
- FBP is unambiguous
  - No “multiple solution” issues
  - Never over- nor under-constrained

But which is faster?

- FBP is always at least a little faster
- CBP solutions have many more parts, hence many more rules to evaluate
  - In the worst typical case, CBP is 3x slower than FBP. YMMV.

Actual Comparison

- 25m high
- 7.75 turns
- Treads are a shared sub-assembly
- TLA not shared – constraints/FBP for each tread)

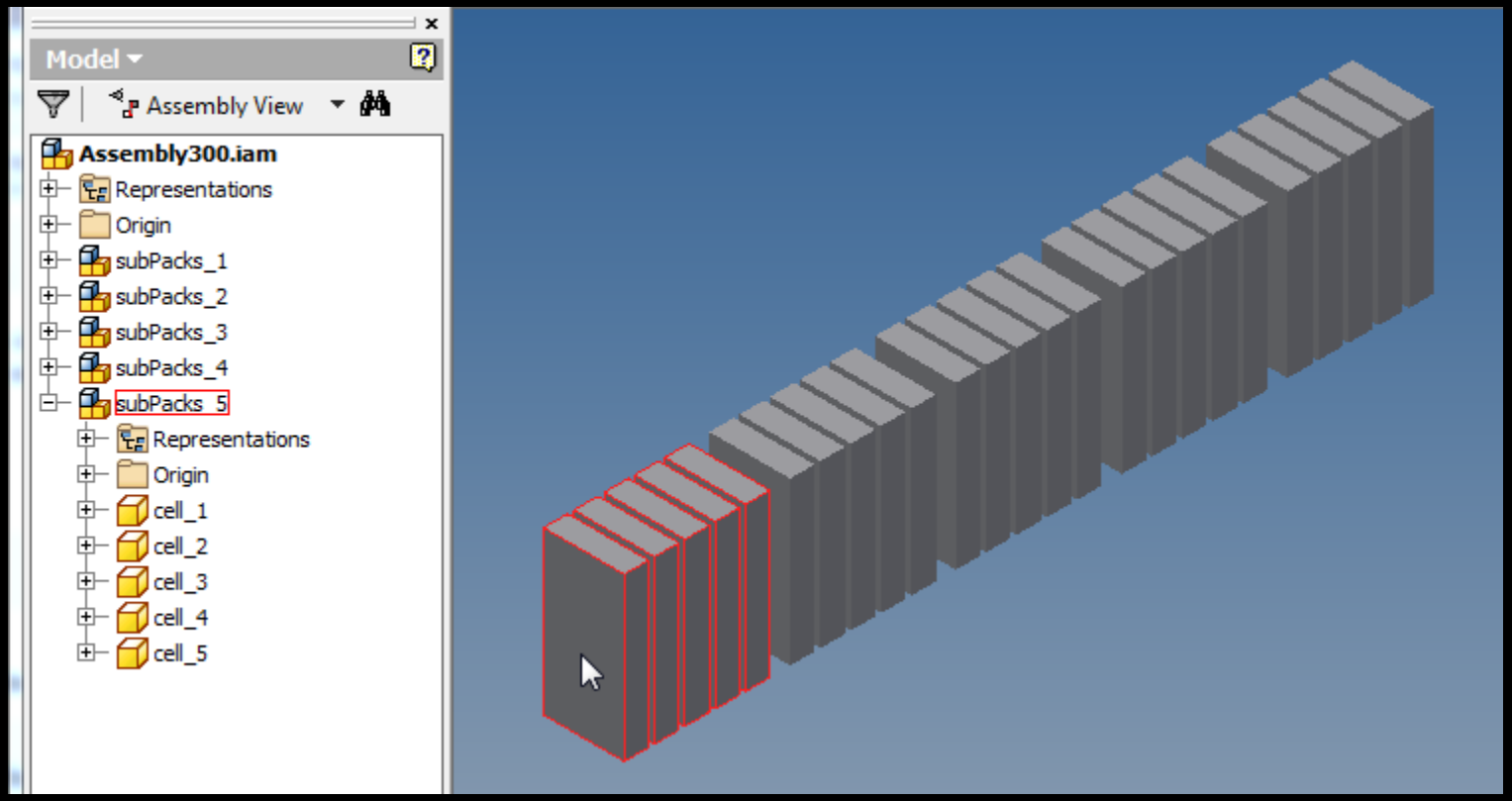


	Time to build	Occurrences	Files	Intent Parts
FBP	0:44	838	12	1679
CBP	1:41	838	12	2517

Shared Assemblies

- "Shared" assemblies minimize computation times
  - Intent keeps track of all assembly participants (occs, constraints, patterns, etc) and caches and reuses member files whenever possible
  - All assembly-modeling is minimized
  - Still must compute the participants that WOULD be needed … this is relatively fast
- Shared assemblies also apply to FBP.
  - Even faster, since there's little or no "would be needed" computations
- Sharing doesn't apply to TLA

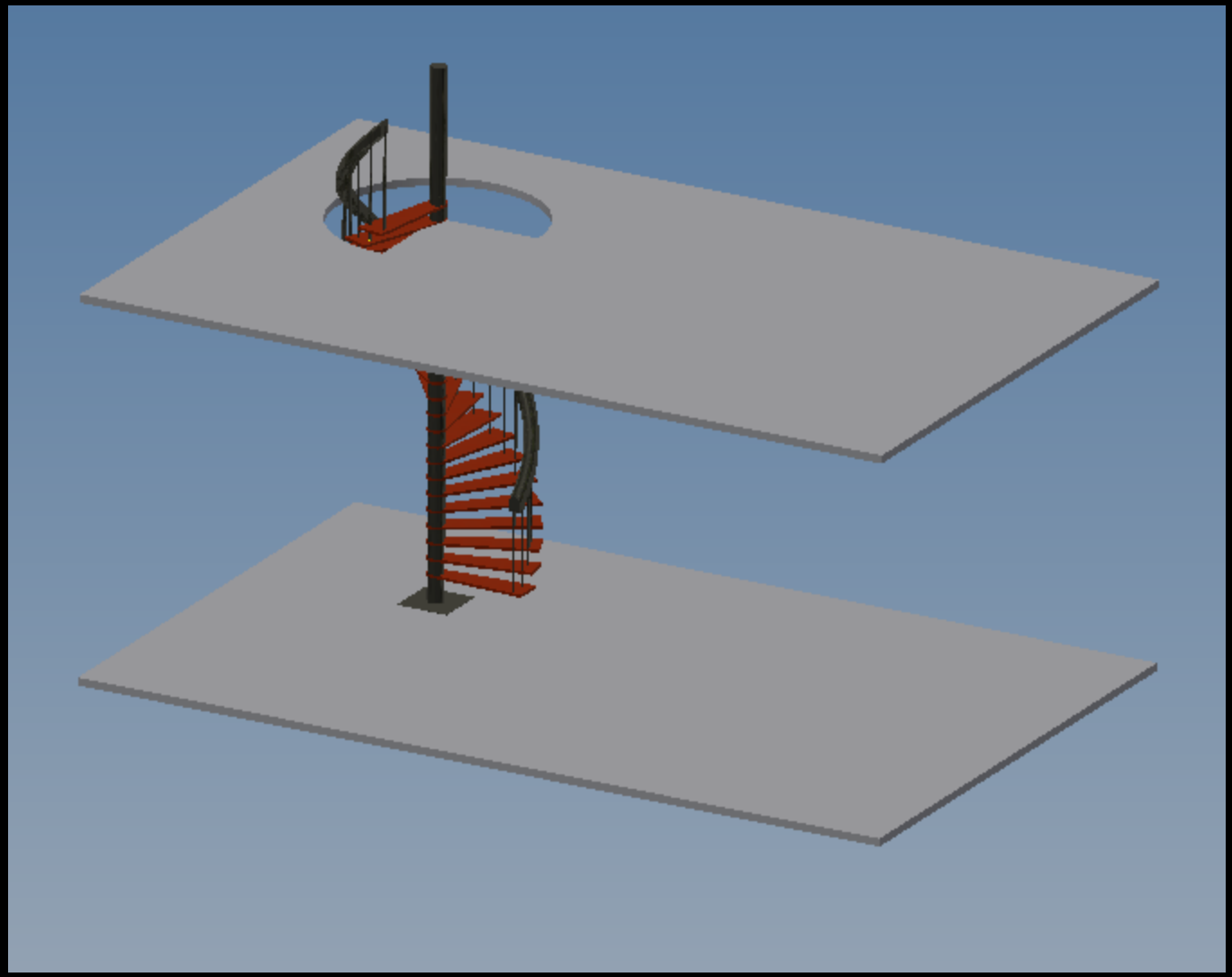
Simplistic Example



Autodesk

Technique	Cells/ Pack	# packs	# Intent parts	Shared time	Un-shared time
CBP	5	100	2100	0:24	1:20
FBP	5	100	603	0:13	0:29
CBP	100	5	2005	0:20	1:00
FBP	100	5	508	0:08	0:16

Better Example



Autodesk

Technique	Sharing	From Size	To Size	Time	Comments
FBP	Yes	3500	35000	1:09	Seems normal
FBP	Yes	35000	3500	0:07	Nice and fast
FBP	Yes	3500	35000	0:35	Seems like it should have been faster
FBP	No	3500	35000	1:05	Same as sharing, OK
FBP	No	35000	3500	0:40	Needs to do a lot of work tearing down previous occurrences
FBP	No	3500	35000	1:00	Same as first time, OK
CBP	Yes	3500	35000	3:45	Quite a bit more than FBP
CBP	Yes	35000	3500	0:08	Nice and fast
CBP	Yes	3500	35000	0:35	Same as FBP, should have been faster
CBP	No	3500	35000	3:48	Same as sharing, OK
CBP	No	35000	3500	2:31	Needs to do even more tearing down than FBP
CBP	No	3500	35000	4:28	Not sure why this is longer than the first time

Summary

- Two different techniques, FBP & CBP
- Each has advantages and disadvantages
- FBP faster than CBP
- Shared assemblies always helps

“On Topic” Q&A

Autodesk

Three Random Tips

1. Use Iv…OccurrencePattern instead of Child-list where possible. *Much faster.* Can only be used where pattern elements are identical, and position of elements is well-defined.

2. GetNewPartNumber. This method is executed after the member file is fully created and updated, but before it is saved. You can use it to do any customization of the member files.

3. Avoid chaining rules to avoid "deep" recursions (e.g., in a Child list, `origin = child.previous.origin+Vector(…))`. Although this seems straightforward and safe, under some circumstances, this kind of rule can cause all the referenced rules to be executed from the same call, resulting in a fatal stack overflow

Q&A – Open Discussion

Autodesk

Thanks!

- Send us suggestions for future topics
- Send us your favorite little “tips”
- See you next time!


Autodesk®