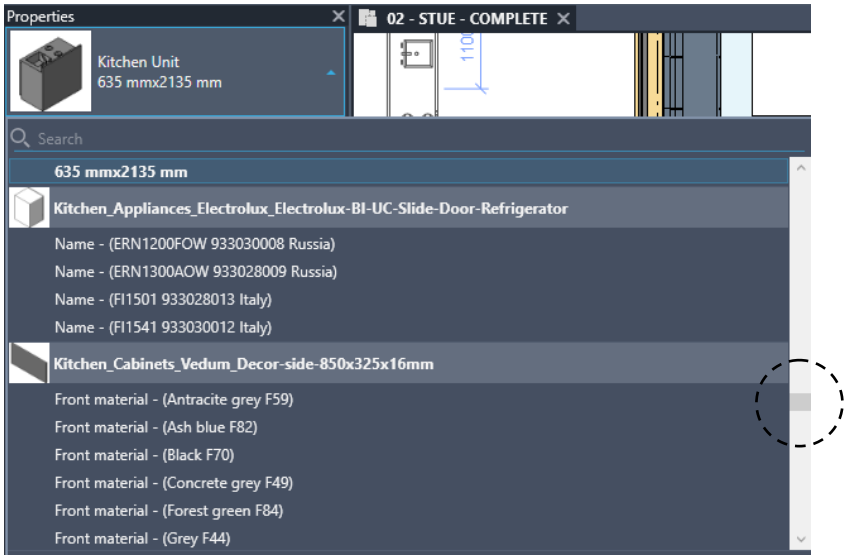
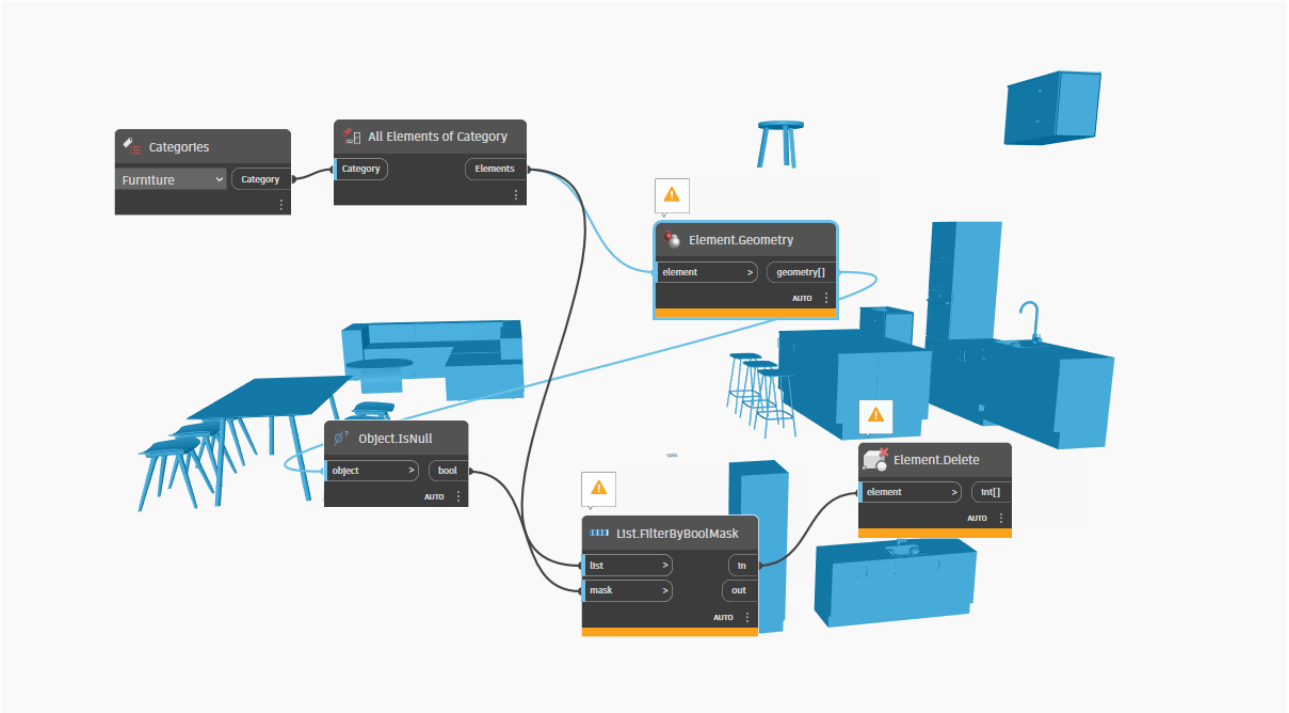
Slette ubrugte Familys

Når man arbejder i større projekter, hvor vi er mange brugere, kan modellen hurtigt blive fyldt med Familys som ikke nødvendigvis bliver brugt. Det kan være alt fra møbler, symboler, døre, VVS-komponenter osv. Som er blevet indlæst, testet og derefter ikke brugt alligevel. Udover at man får ryddet op og et bedre overblik, så bliver Revit-filen også hurtigere ift. hastighed, indlæsningstid og synkronisering.

Dette har gjort jeg har undersøgt om det ikke var muligt at lave et script der fjerner de her ubrugte Familys.



Først prøvede jeg mig lidt frem ved at fjerne ubrugte Familys ved hjælpe af Dynamo-noder. Målet var at identificere hvilke Familys der ikke var placeret i modellen og derefter slette dem automatisk.

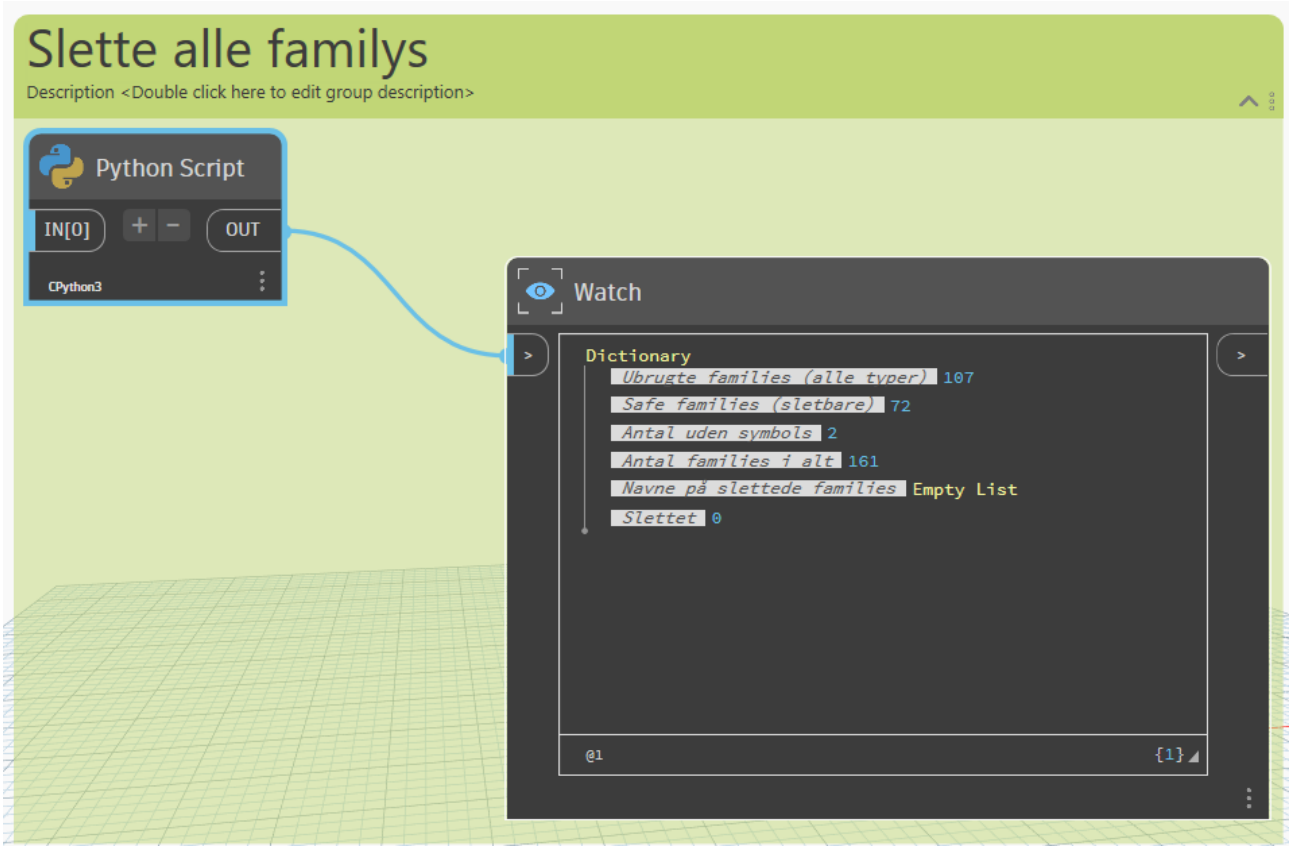


Men dette kunne ikke lade sig gøre med standardnoder i Dynamo. Blandt andet fordi dynamo ikke har en node der kan slette elementer direkte fra projektet. Der findes ikke en parameter eller metode der viser om en Family er placeret eller ej.

Da det ikke lykkedes med de almindelige noder, valgte jeg at brugte Python Scripting i Dynamo i stedet. Her fik jeg hjælp af ChatGPT som guidede mig til at skrive et Python-Script der:

- Går igennem alle Familys i projektet.
- Undersøger om de er i brug.
- Sletter dem automatisk hvis de ikke er i brug.

Dette gjorde det muligt at automatisere oprydning af modellen uden at slutte slette Familys manuelt en ad gangen.




```
Import af nødvendige biblioteker
import clr
clr.AddReference('RevitServices')
from RevitServices.Persistence import DocumentManager
doc = DocumentManager.Instance.CurrentDBDocument

clr.AddReference("RevitAPI")
from Autodesk.Revit.DB import (
   FilteredElementCollector, Family, FamilySymbol,
   ElementType, Transaction, BuiltInCategory, ElementId, StorageType
)
```

Her importeres .NET-biblioteker og Revit API-klasser, som bruges til at hente data og ændre i Revit-modellen.

```
Find alle families og symbols
all_families = FilteredElementCollector(doc).OfClass(Family).ToElements()
all_symbols = FilteredElementCollector(doc).OfClass(FamilySymbol).ToElements()
```

Scriptet starter med at hente en liste over alle de familier (f.eks. møbler, symboler, objekter), der ligger i projektet.

Find symbols i brug via instanser

```
used_symbol_ids = set()
instances = FilteredElementCollector(doc).WhereElementIsNotElementType().ToElements()
for inst in instances:
  if hasattr(inst, 'Symbol') and inst.Symbol:
    used_symbol_ids.add(inst.Symbol.Id)
```

Derefter kigger det på hele modellen og ser, om nogen af disse familier er placeret et sted i bygningen. Dem der ikke bliver brugt, går videre til næste trin.

Step 3: Find symbols brugt i typeparametre

```
types = FilteredElementCollector(doc).WhereElementIsElementType().ToElements()
for type_elem in types:
  for param in type_elem.Parameters:
    if param.StorageType == StorageType.ElementId:
      try:
        ref_id = param.AsElementId()
        if isinstance(ref_id, ElementId) and ref_id.IntegerValue != -1:
          used_symbol_ids.add(ref_id)
      except:
        pass
```

Nogle familys bruges som en indstilling i noget andet (fx et greb i en dørtype), selvom de ikke er synlige. Det tjekker scriptet også, så man ikke kommer til at slette noget vigtigt ved en fejl.

Find families hvor INGEN symbols er i brug

```
unused_families = []
no_symbol_families = 0

for fam in all_families:
  symbol_ids = fam.GetFamilySymbolIds()
  if not symbol_ids:
    no_symbol_families += 1
    continue
  if not any(sym_id in used_symbol_ids for sym_id in symbol_ids):
    unused_families.append(fam)
```

Når den har fundet dem, der hverken er synlige eller bruges 'skjult', laver den en liste over dem, der trygt kan slettes.

Beskyt systemfamilier og tags

```
excluded_categories = {
  int(BuiltInCategory.OST_RoomTags),
  int(BuiltInCategory.OST_DoorTags),
  int(BuiltInCategory.OST_WindowTags),
  int(BuiltInCategory.OST_WallTags),
```

```
    int(BuiltInCategory.OST_FloorTags),
    int(BuiltInCategory.OST_CeilingTags),
    int(BuiltInCategory.OST_MaterialTags),
    int(BuiltInCategory.OST_MultiCategoryTags),
    int(BuiltInCategory.OST_KeynoteTags),
    int(BuiltInCategory.OST_SectionHeads),
    int(BuiltInCategory.OST_ElevationMarks),
    int(BuiltInCategory.OST_Views),
    int(BuiltInCategory.OST_Levels),
    int(BuiltInCategory.OST_Grids),
    int(BuiltInCategory.OST_Dimensions),
    int(BuiltInCategory.OST_Railings),
    int(BuiltInCategory.OST_Stairs),
    int(BuiltInCategory.OST_Walls),
    int(BuiltInCategory.OST_Floors),
    int(BuiltInCategory.OST_Ceilings),
    int(BuiltInCategory.OST_Rooms),
    int(BuiltInCategory.OST_Viewports),
    int(BuiltInCategory.OST_ProfileFamilies),
    int(BuiltInCategory.OST_StairsRailing),
    10205,  # ReferencePlanes
    297    # Balusters
}
safe_families = []
for fam in unused_families:
    cat = fam.FamilyCategory
    if cat and hasattr(cat, 'Id'):
        if cat.Id.IntegerValue not in excluded_categories:
            if not fam.Name.startswith("VIA_"):
                safe_families.append(fam)
```

Scriptet er lavet til at springe over ting, der er nødvendige for projektet – fx tags, mål, vægge og alt der starter med 'VIA_'.

Sletning

```
t = Transaction(doc, "Delete unused loadable families")
t.Start()
deleted_names = []
for fam in safe_families:
    try:
        doc.Delete(fam.Id)
        deleted_names.append(fam.Name)
    except:
        pass
t.Commit()
```

En transaktion startes og alle safe_families slettes. Navnene gemmes til output.

Til sidst sletter den alt, der er sikkert at fjerne, uden du behøver gøre det manuelt. Den viser dig bagefter, hvad der blev slettet.

Ulemper:

Selvom scriptet fungerer, er der nogle væsentlige ulemper og risici man skal være opmærksom på.

1. Scriptet kan slette mere end forventet.

Min egen erfaring slettede scriptet Tags og andre annotative Familys som jeg troede var i brug men som revit ikke registrerede som placeret. Dette viser at der er en risiko for at fjerne vigtige elementer.

2. Ingen fortryd-knap

Når man sletter noget med Python i Dynamo sker det direkte i modellen og kan derfor ikke fortrydes. Man kan nogle gange ”Undo” i Revit, men det ikke altid det virker efter en Dynamo-Kørsel

Konklusion

Der er automatiseret en ellers manuel og tidskrævende proces: oprydning af ubrugte familys i Revit-Modellen. Hvilket forbedrer ydeevnen, give bedre overblik og gør det lettere at arbejde i projektet. Scriptet fungerer effektivt og sparer tid til manuel sletning, dog har jeg erfaret at man skal bruge det med forsigtighed, da det kan slette elementer som man troede var i brug.