

Design Automation for Revit: Basics and beyond

Rahul Bhoje

Senior Principle Engineer | Autodesk, Inc

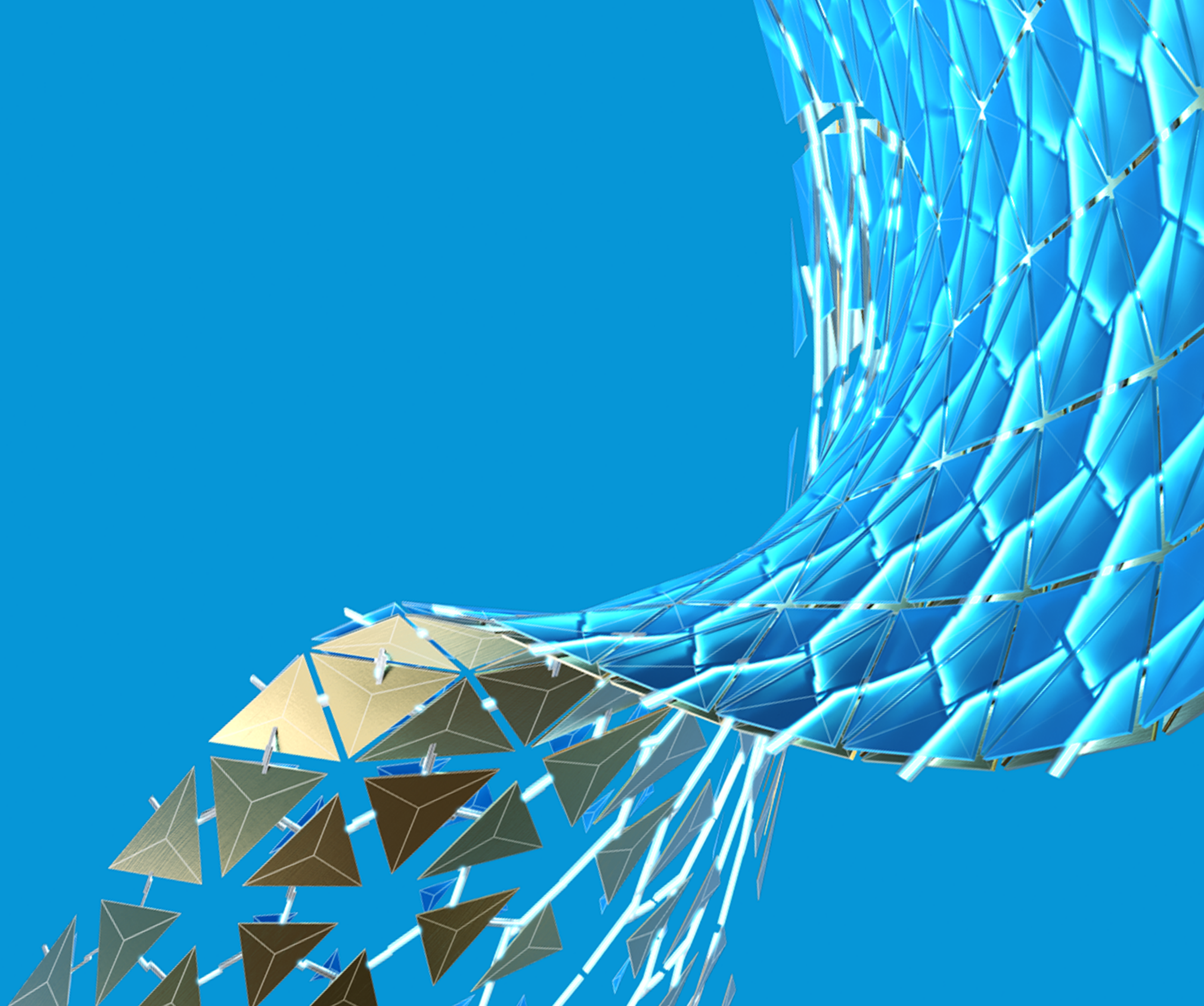


Rahul Bhoje

Senior Principal Software Engineer, Autodesk

Rahul Bhoje is a Senior Principal Software Developer for Revit at Autodesk. He holds a B. Tech degree in Naval Architecture from IIT Madras and a masters degree in Software Systems. He has been with Autodesk for 13 years working on several features of Revit. Currently he is working on developing Design Automation for Revit. In 2008, he started working on Revit Conceptual Modeling design where he implemented Adaptive Components, Point Elements and Divided Surface. He has then worked on several Revit features like Family, Groups, Links and Worksharing. He has a total of 21 years of CAD/CAM development experience.

Agenda

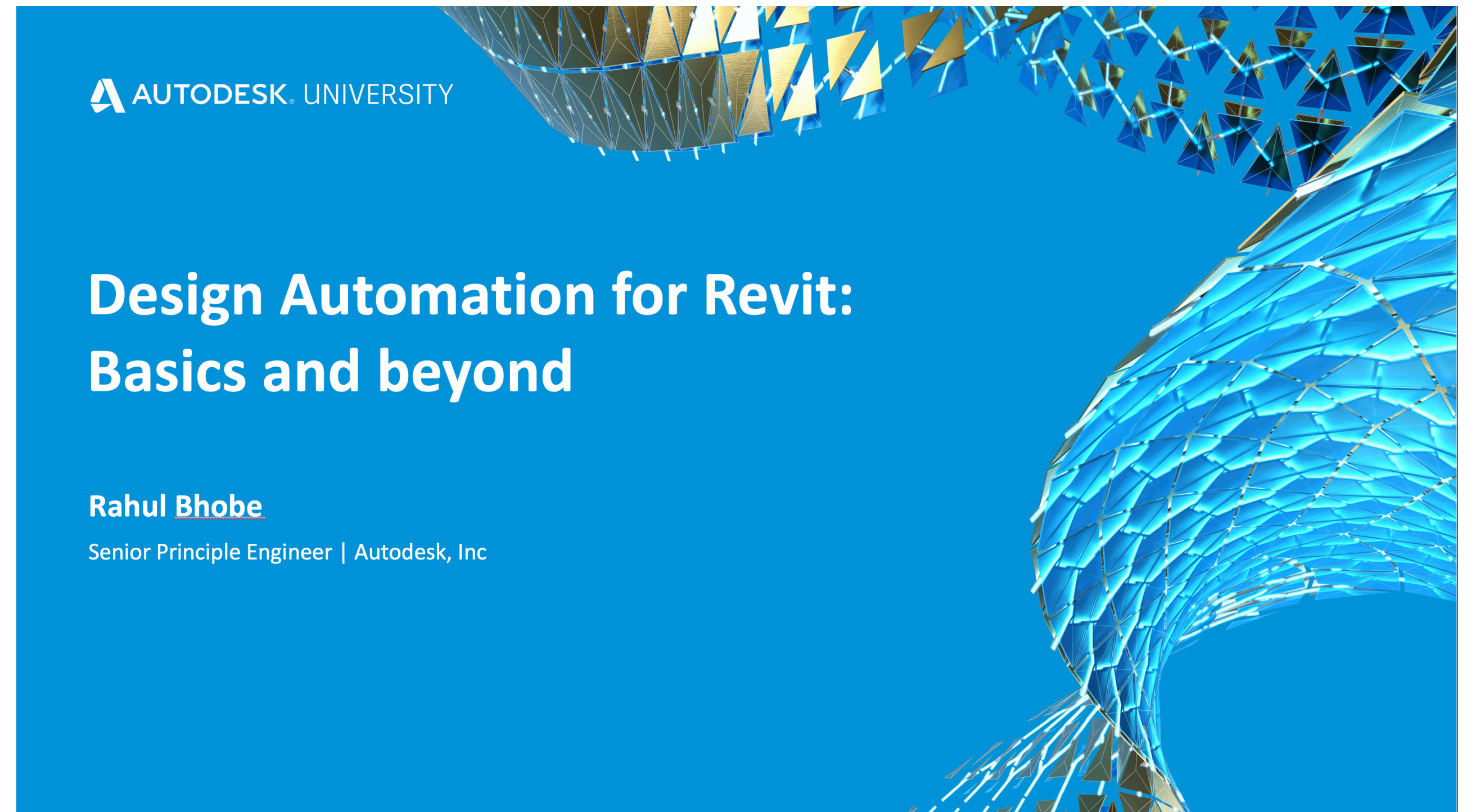


Agenda

1. Design Automation Overview
2. Delete Walls
3. Count It
4. Sketch It
5. Stadium Panels
6. Grid Object Placement

アジェンダ

- Design Automation とは
- Delete Walls
- Count It
- Sketch It
- Stadium Panels
- Grid Object Placement

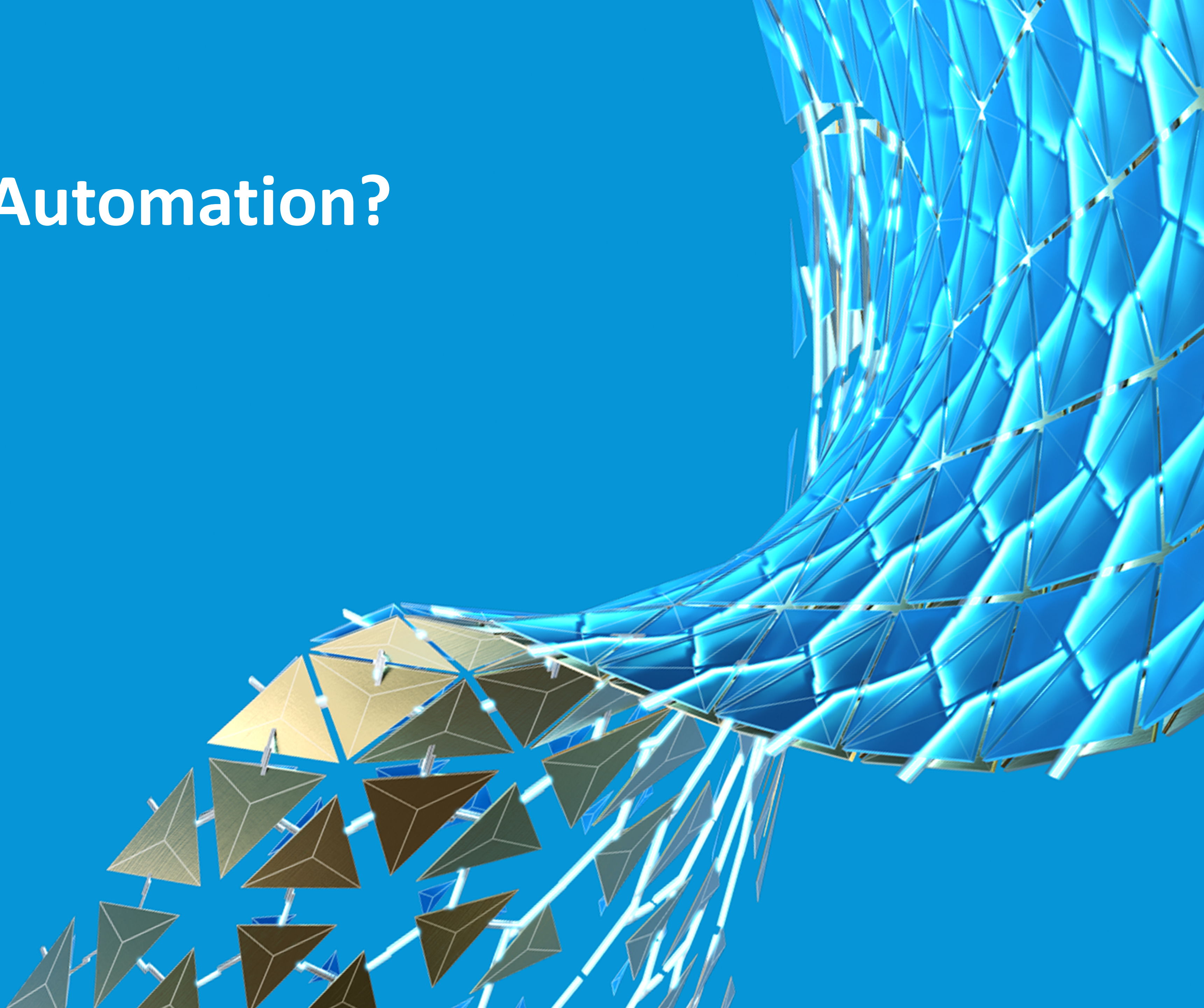


小笠原 龍司 (Ryuji Ogasawara)

2014年からオートデスクにて、Revit や Navisworks 等、デスクトップ製品 API・アドイン開発・API のサポート、及び Forge プラットフォーム API のエバンジェリストとして活動中。

前職では、Webアプリケーションエンジニアとして環境制御システムやリアルタイムコミュニケーションシステム、統合基幹業務システムの開発に従事。

What is Design Automation?





Design Automation API v3

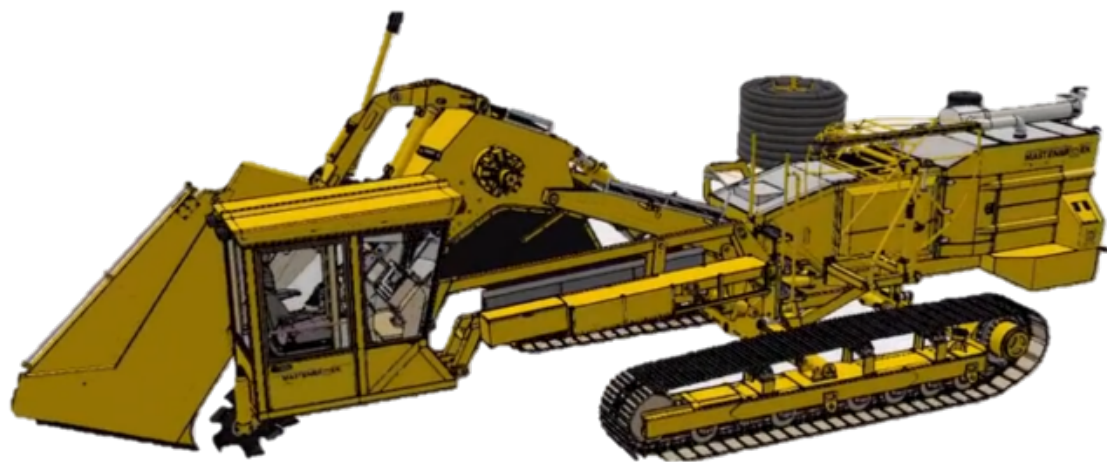
Run automations on your design files



AUTODESK® REVIT®



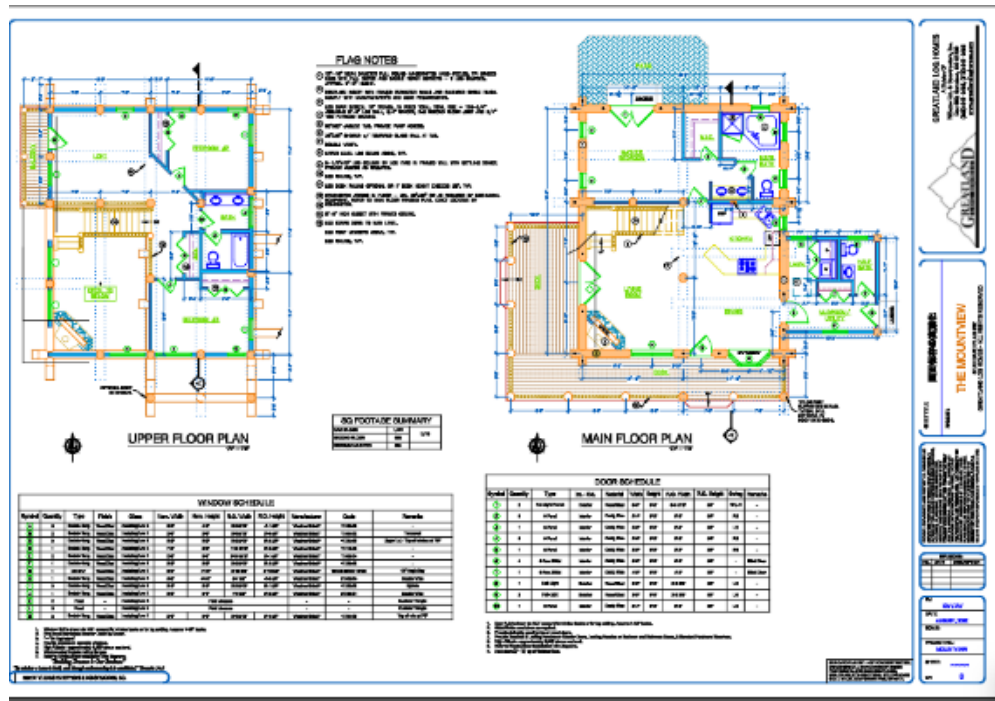
AUTODESK® INVENTOR®

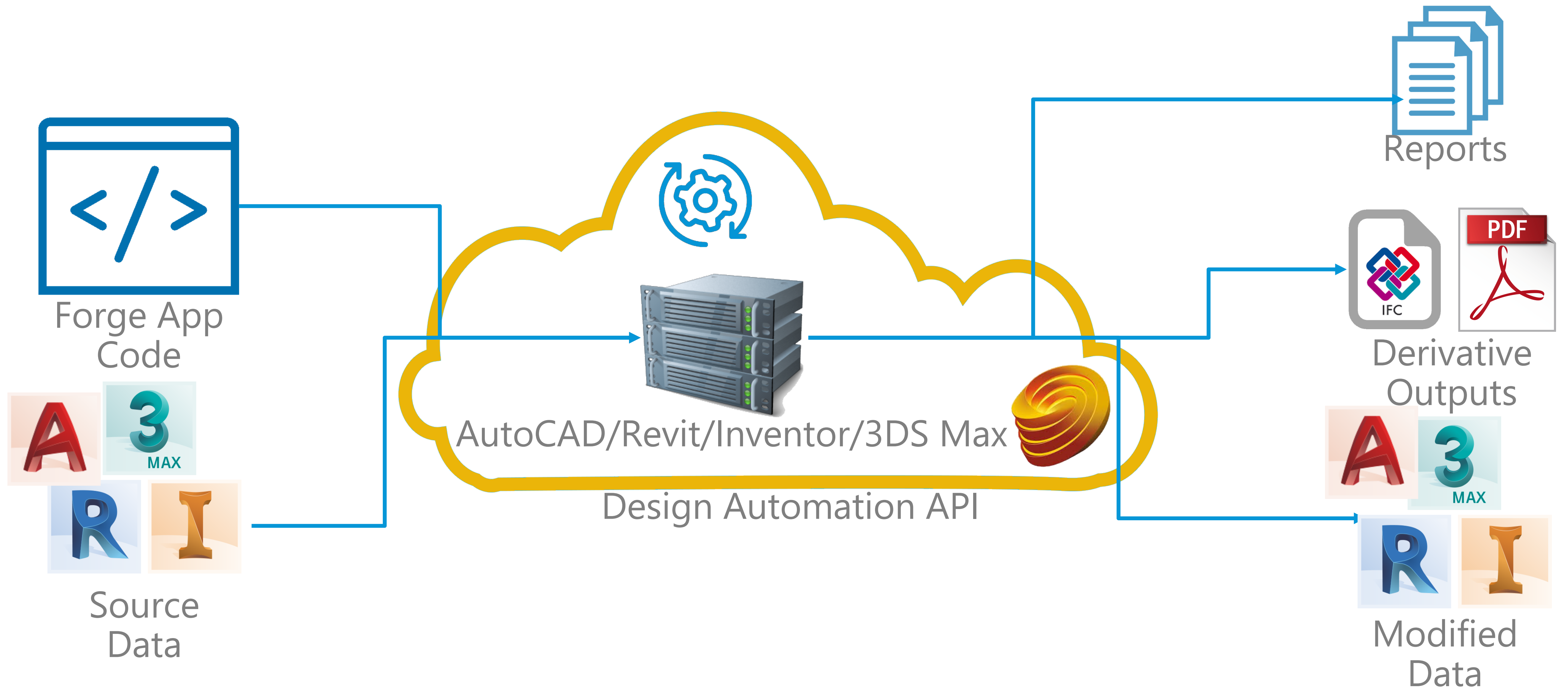


AUTODESK® 3DS MAX®

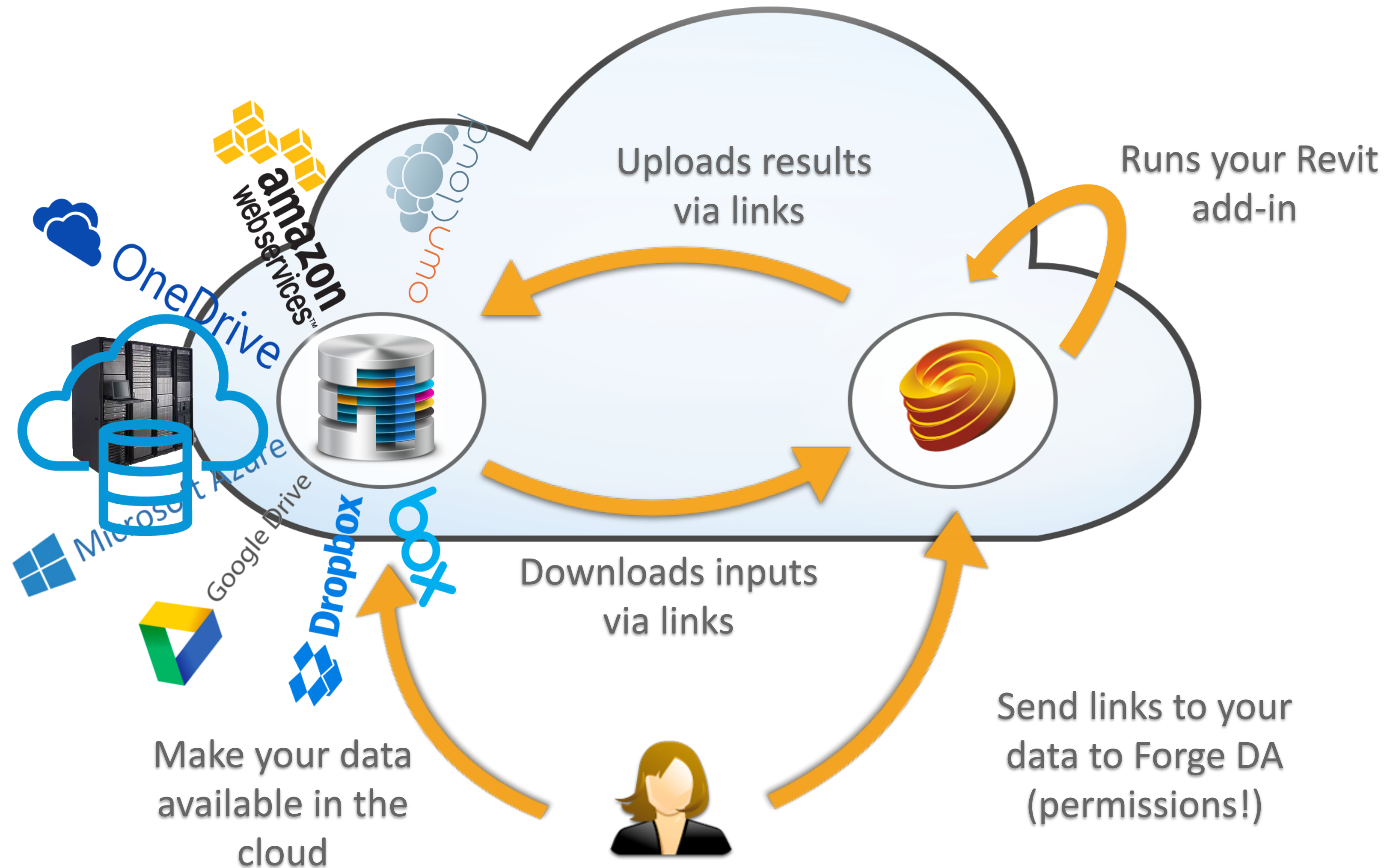


AUTODESK® AUTOCAD®





How Does It Work?



i Input/outputs are **NOT** stored, they are downloaded/processed and **DISCARDED**

The API

V3 HTTPS Endpoint	Programming concept	Product concept
/workitems	Function call	Product execution, job
/activities	Function definition	Input/Output types, parameters
/appbundles	Shared library	Addin
/engines	Instruction set	Product exe (Revit)

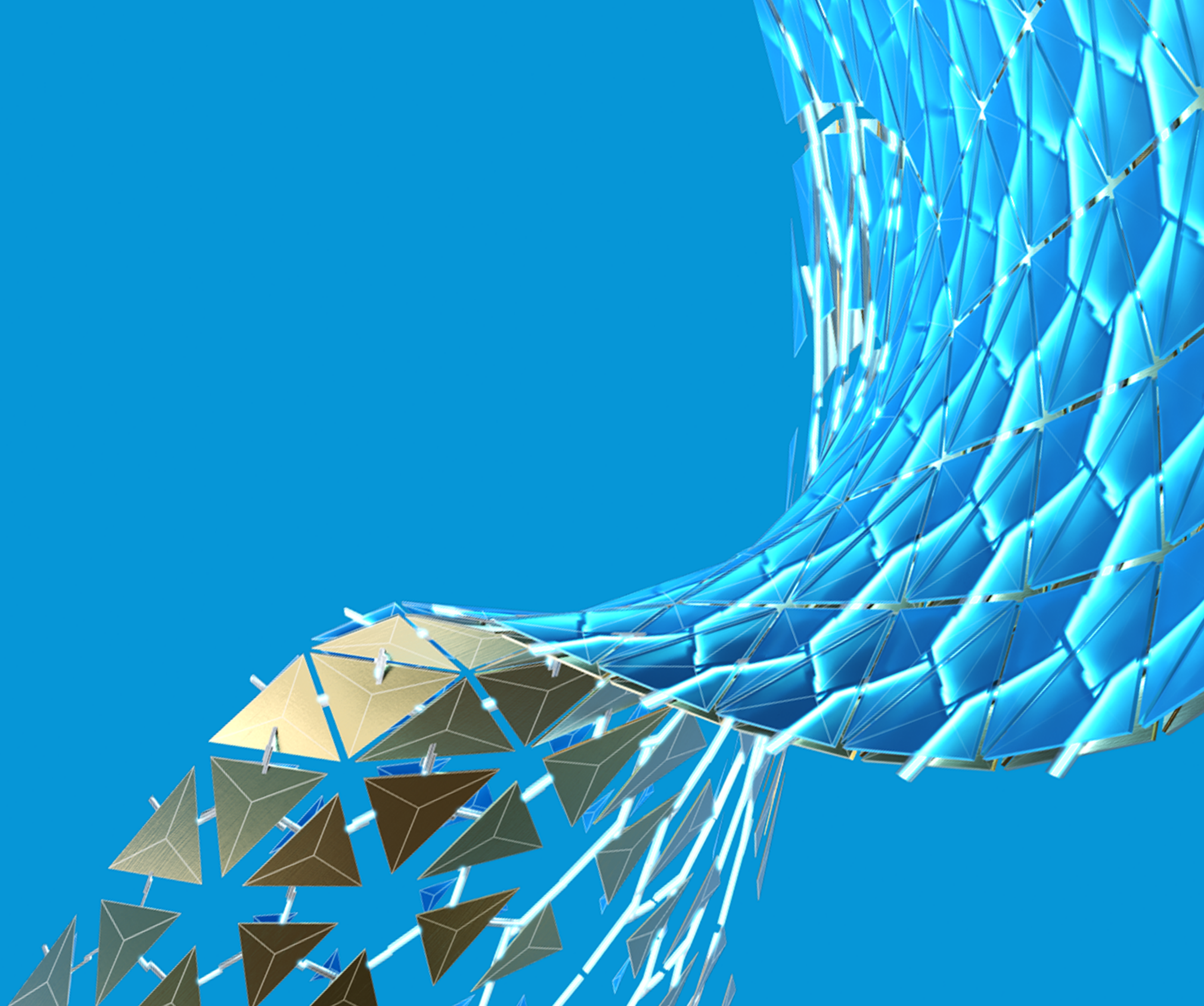
Function Analogy

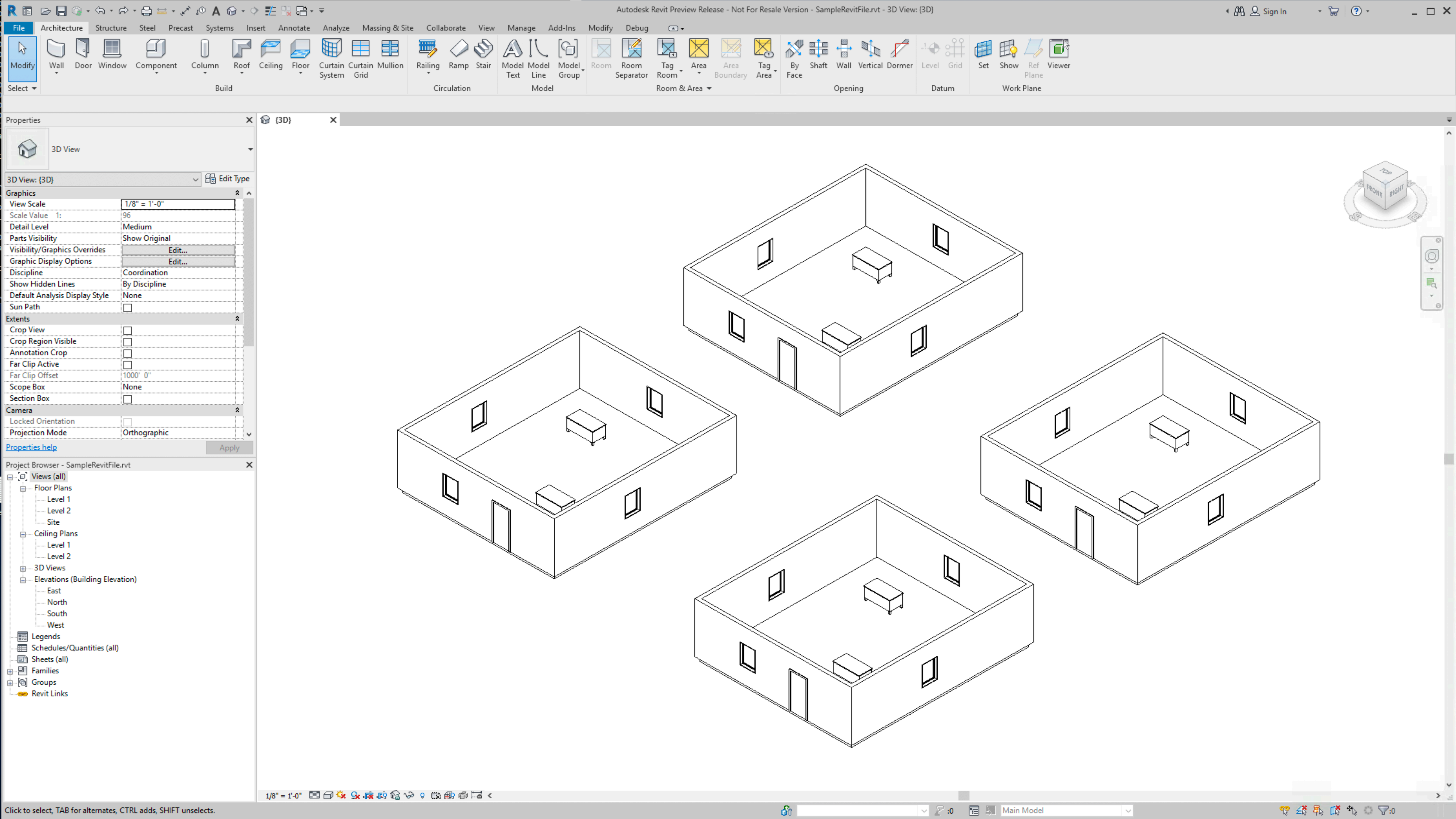
```
static void Activity(int first, int second, out int sum, out int product)
{
    sum = first + second;
    Console.WriteLine($"The sum of {first} and {second} is {sum}");
    product = first * second;
    Console.WriteLine($"The product of {first} and {second} is {product}");
}

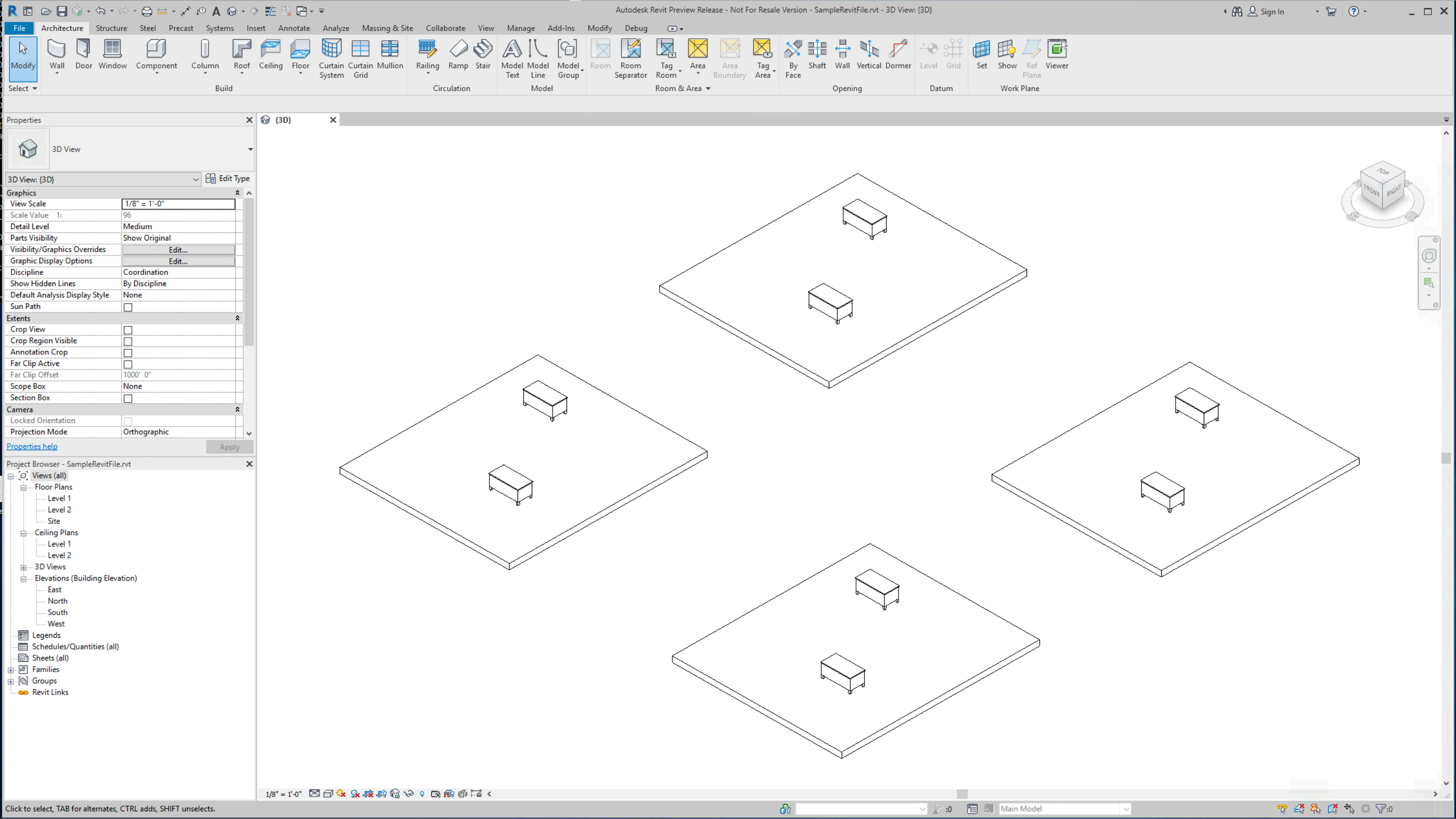
static void Work()
{
    // Workitem 1
    int sum, product;
    Activity(first: 13, second: 17, sum: out sum, product: out product);

    // Workitem 2
    int another_sum, another_product;
    Activity(first: 23, second: 31, sum: out another_sum, product: out another_product);
}
```


Delete Walls







Delete Walls (activity / command line)

```
{
  "id": "DeleteWallsActivity",
  "commandLine": [ '$(engine.path)\\revitcoreconsole.exe' /i "$(args[rvtfile].path)" /al "$(appbundles[DeleteWallsBundle].path)" ],
  "parameters": {
    "rvtfile": {
      "zip": false,
      "ondemand": false,
      "verb": "get",
      "description": "input revit model",
      "required": true
    },
  },
  "result": {
    "zip": false,
    "ondemand": false,
    "verb": "put",
    "description": "results",
    "required": true,
    "localName": "result.rvt"
  },
},
"engine": "Autodesk.Revit+2021",
"appbundles": [ "MyNickname.DeleteWallsBundle+myalias" ],
"description": "Delete walls from Revit file."
}
```

/al "\$(appbundles[DeleteWallsBundle].path)"

\$(engine.path)\\revitcoreconsole.exe

/i "\$(args[rvtfile].path)"

["MyNickname.DeleteWallsBundle+myalias"]

Delete Walls (activity / parameters)

```
{
  "id": "DeleteWallsActivity",
  "commandLine": [ '$(engine.path)\\revitcoreconsole.exe /i "$(args[rvtFile].path)" /al "$(appbundles[DeleteWallsBundle].path)"' ],
  "parameters": {
    "rvtfile": {
      "zip": false,
      "ondemand": false,
      "verb": "get",
      "description": "input revit model",
      "required": true
    },
    "result": {
      "zip": false,
      "ondemand": false,
      "verb": "put",
      "description": "results",
      "required": true,
      "localName": "result.rvt"
    }
  },
  "engine": "Autodesk.Revit+2021",
  "appbundles": [ "MyNickname.DeleteWallsBundle+myalias" ],
  "description": "Delete walls from Revit file."
}
```

rvtfile

"verb": "get"

"required": true

result

"verb": "put"

"localName": "result.rvt"

"required": true

Delete Walls (workitem)

```
{
  "activitId": "MyNickname.DeleteWallsActivity+myalias",
  "arguments": {
    "rvtfile": {
      "verb": "get",
      "url": "https://my-storage-location.com/input.rvt"
    },
    "result": {
      "verb": "put",
      "url": "https://my-storage-location.com/output.rvt"
    }
  }
}
```


Delete Walls (workitem)

```
{
  "activitId": "MyNickname.DeleteWallsActivity+myalias",
  "arguments": {
    "rvtfile": {
      "verb": "get",
      "url": "https://my-storage-location.com/another_input.rvt"
    },
    "result": {
      "verb": "put",
      "url": "https://my-storage-location.com/another_output.rvt"
    }
  }
}
```


Delete Walls (workitem)

```
{
  "activitId": "MyNickname.DeleteWallsActivity+myalias",
  "arguments": {
    "rvtfile": {
      "verb": "get",
      "url": "https://developer-dev.api.autodesk.com/oss/v2/signedresources/a407d7d6-f32a-4df8-b772-824c4440eb41?region=US"
    },
    "result": {
      "verb": "put",
      "url": "https://developer-dev.api.autodesk.com/oss/v2/signedresources/b5d103c3-cd80-415f-bd59-fe17a1cbb7b8?region=US"
    }
  }
}
```

Delete Walls (appbundle)

```
public class DeleteWallsApp : IExternalDBApplication
{
    public ExternalDBApplicationResult OnStartup(ControlledApplication app)
    {
        DesignAutomationBridge.DesignAutomationReadyEvent += HandleDesignAutomationReadyEvent;
        return ExternalDBApplicationResult.Succeeded;
    }
    public ExternalDBApplicationResult OnShutdown(ControlledApplication app)
    {
        return ExternalDBApplicationResult.Succeeded;
    }
    public void HandleDesignAutomationReadyEvent(object sender, DesignAutomationReadyEventArgs e)
    {
        DeleteAllWalls(e.DesignAutomationData);
        e.Succeeded = true;
    }
}
```


Delete Walls (appbundle)

```
public static void DeleteAllWalls(DesignAutomationData data)
{
    Document doc = data.RevitDoc;

    using (Transaction transaction = new Transaction(doc))
    {
        FilteredElementCollector col = new FilteredElementCollector(doc).OfClass(typeof(Wall));
        transaction.Start("Delete All Walls");
        doc.Delete(col.ToElementIds());
        transaction.Commit();
        Console.WriteLine($"Deleted {col.GetElementCount()} Walls");
    }

    doc.SaveAs("result.rvt");
}
```

Delete Walls (Recap / Learnings)

Activity definition and details

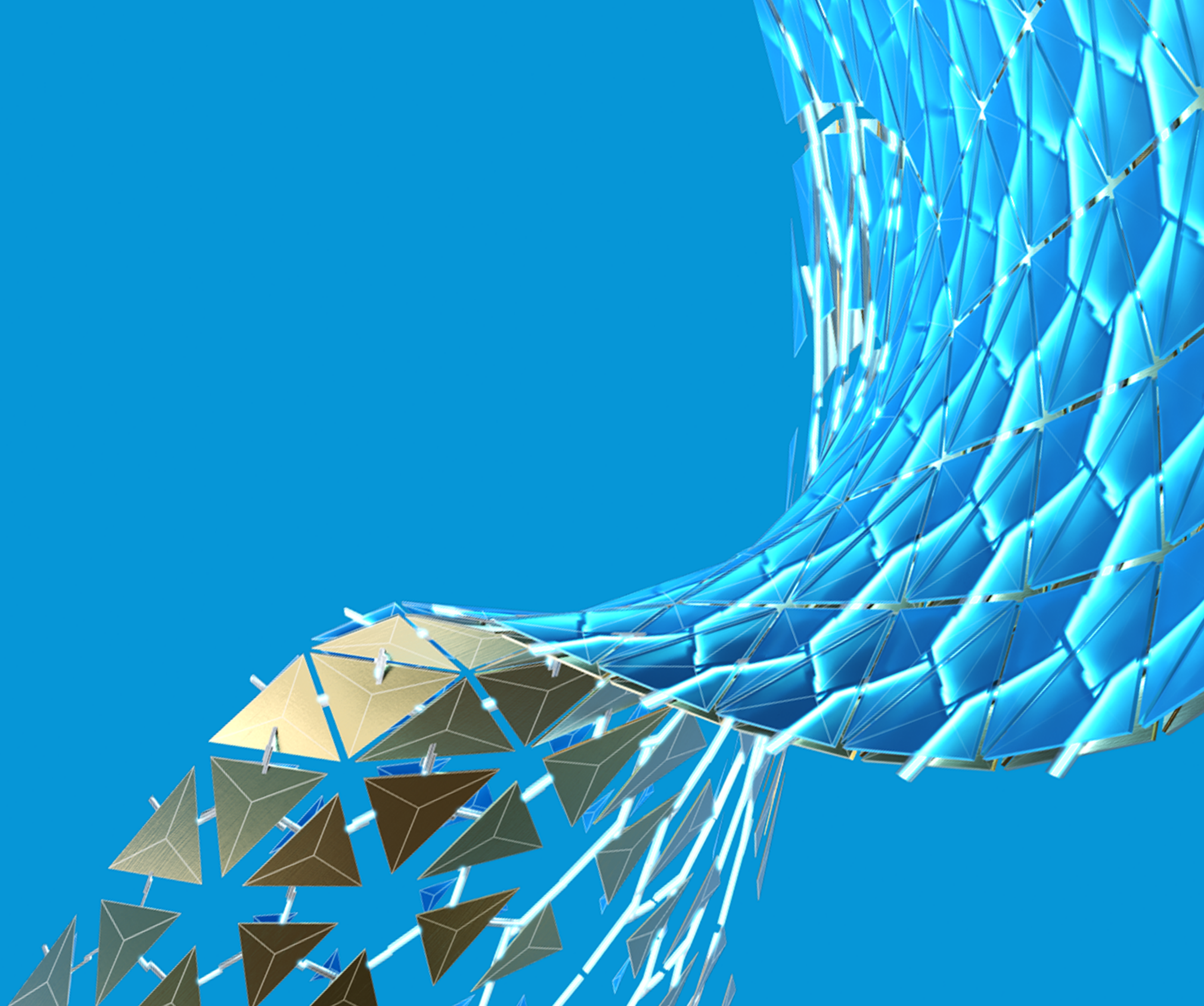
Workitem and relationship with activity

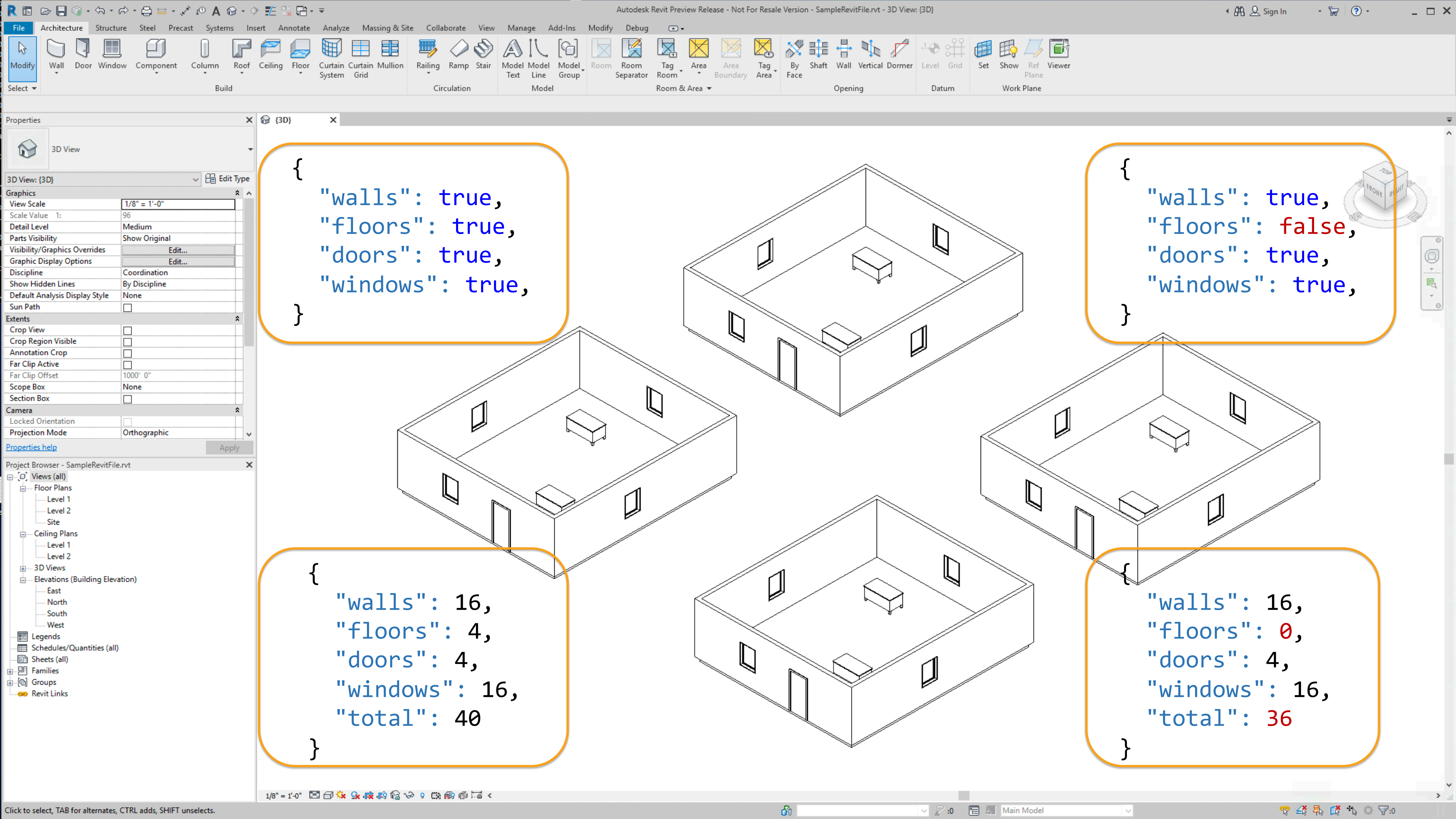
Design Automation event handler in appbundle

Input Revit Document in appbundle

Basic application with modification workflow

CountIt





```
{  
  "walls": true,  
  "floors": true,  
  "doors": true,  
  "windows": true,  
}
```

```
{  
  "walls": true,  
  "floors": false,  
  "doors": true,  
  "windows": true,  
}
```

```
{  
  "walls": 16,  
  "floors": 4,  
  "doors": 4,  
  "windows": 16,  
  "total": 40  
}
```

```
{  
  "walls": 16,  
  "floors": 0,  
  "doors": 4,  
  "windows": 16,  
  "total": 36  
}
```


CountIt (activity / command line)

```
{
  "id": "CountItActivity",
  "commandLine": [ '$(engine.path)\\revitcoreconsole.exe /i "$(args[rvtfile].path)" /al "$(appbundles[CountItBundle].path)" ' ],
  "parameters": {
    "rvtfile": {
      "verb": "get",
      "required": true
    },
    "params": {
      "verb": "get",
      "required": false,
      "localName": "params.json"
    },
    "result": {
      "verb": "post",
      "required": true,
      "localName": "result.json"
    }
  },
  "engine": "Autodesk.Revit+2021",
  "appbundles": [ "MyNickname.CountItBundle+myalias" ],
  "description": "Count Elements in Revit file."
}
```

/al "\$(appbundles[CountItBundle].path)"

["MyNickname.CountItBundle+myalias"]

CountIt (activity / parameters)

```
{
  "id": "CountItActivity",
  "commandLine": [ '$(engine.path)\\revitcoreconsole.exe /i "$(args[rvtFile].path)" /al "$(appbundles[CountItBundle].path)" ' ],
  "parameters": {
    "rvtfile": {
      "verb": "get",
      "required": true
    },
    "params": {
      "verb": "get",
      "required": false,
      "localName": "params.json"
    },
    "result": {
      "verb": "post",
      "required": true,
      "localName": "result.json"
    }
  },
  "engine": "Autodesk.Revit+2021",
  "appbundles": [ "MyNickname.CountItBundle+myalias" ],
  "description": "Count Elements in Revit file."
}
```

```
{
  "walls": true,
  "floors": false,
  "doors": true,
  "windows": true
}
```

```
{
  "walls": 16,
  "floors": 0,
  "doors": 4,
  "windows": 16,
  "total": 36
}
```


CountIt (appbundle)

```
using Newtonsoft.Json;
```

```
class CountItParams
```

```
{
```

```
    public bool walls { get; set; } = false;  
    public bool floors { get; set; } = false;  
    public bool doors { get; set; } = false;  
    public bool windows { get; set; } = false;
```

```
    static public CountItParams Parse()
```

```
{
```

```
    if (!File.Exists("params.json"))  
        return new CountItParams { walls = true, floors = true, doors = true, windows = true };
```

```
    string jsonContents = File.ReadAllText("params.json");  
    return JsonConvert.DeserializeObject<CountItParams>(jsonContents);
```

```
}
```

```
}
```

CountIt (appbundle)

```
void CountElements(Document revitDoc, CountItParams countItParams, ref CountItResults results)
{
    if (countItParams.walls)
    {
        FilteredElementCollector elemCollector = new FilteredElementCollector(revitDoc);
        elemCollector.OfClass(typeof(Wall));
        int count = elemCollector.ToElementIds().Count;
        results.walls += count;
        results.total += count;
    }

    if (countItParams.floors)
    {
        .
        .
    }
}
```


CountIt (workitem)

```
{  
  "activitId": "MyNickname.CountItActivity+myalias",  
  "arguments": {  
    "rvtfile": {  
      "url": "https://my-storage-location.com/input.rvt"  
    },  
    "params": {  
      "url": "data:application/json,{"walls": true, "floors": true, "doors": true}"  
    },  
    "result": {  
      "verb": "put",  
      "url": "https://my-storage-location.com/output.json"  
    }  
  }  
}
```

```
{  
  "walls": true,  
  "floors": true,  
  "doors": true  
}
```

CountIt (Recap / Learnings)

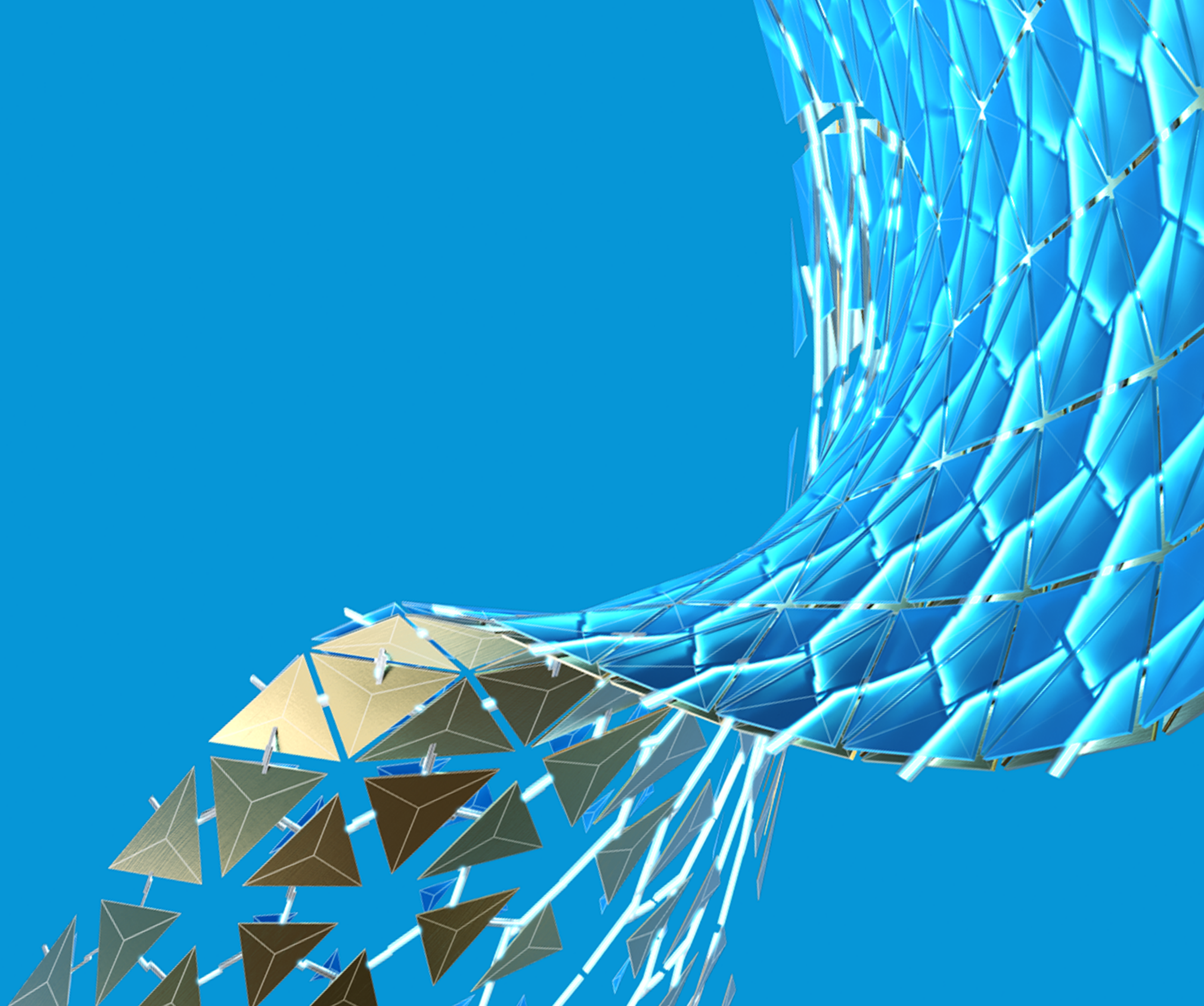
Parametrize Design Automation jobs

Default value for parameters

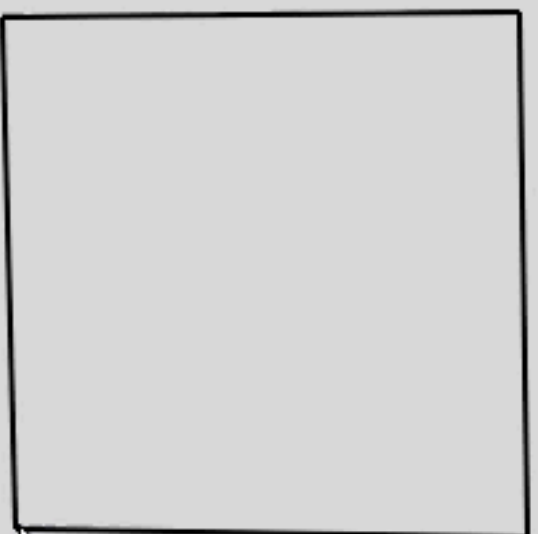
Generate text/json output files

Application with **extraction** workflow

SketchIt



Sketch It



Generate Revit model

SketchIt (activity / command line)

```
{
  "id": "SketchItActivity",
  "commandLine": [ '$(engine.path)\\revitcoreconsole.exe /i "$(args[rvtfile].path)" /al "$(appbundles[SketchItBundle].path)" ' ],
  "parameters": {
    "sketchItInput": {
      "verb": "get",
      "required": false,
      "localName": "SketchItInput.json"
    },
    "result": {
      "verb": "post",
      "required": true,
      "localName": "result.rvt"
    }
  },
  "engine": "Autodesk.Revit+2021",
  "appbundles": [ "MyNickname.SketchItBundle+myalias" ],
  "description": "Sketch on browser to generate Revit file."
}
```

SketchIt (activity / command line)

```
{  
  "id": "SketchItActivity",  
  "commandLine": [ '$(engine.path)\\revitcoreconsole.exe /a1 "$(appbundles[SketchItBundle].path)"' ],  
  "parameters": {  
    "sketchItInput": {  
      "verb": "get",  
      "localName": "SketchItInput.json" /a1 "$(appbundles[SketchItBundle].path)"  
    },  
    "result": {  
      "verb": "post",  
      "localName": "result.rvt"  
    }  
  },  
  [ "MyNickname.SketchItBundle+myalias" ]  
  "engine": "Autodesk.Revit+2021",  
  "appbundles": [ "MyNickname.SketchItBundle+myalias" ],  
  "description": "Sketch on browser to generate Revit file."  
}
```


SketchIt (activity / parameters)

```
{  
  "id": "SketchItActivity",  
  "commandLine": [ '$(engine.path)\\revitcoreconsole.exe /al "$(appbundles[SketchItBundle].path)" ' ],  
  "parameters": {  
    "sketchItInput": {  
      "verb": "get",  
      "localName": "SketchItInput.json"  
    },  
    "result": {  
      "verb": "post",  
      "localName": "result.rvt"  
    }  
  },  
  "engine": "Autodesk.Revit+2021",  
  "appbundles": [ "MyNickname.SketchItBundle+myalias" ],  
  "description": "Sketch on browser to generate Revit file."  
}
```

SketchIt (appbundle)

```
class Point
{
    [JsonProperty(PropertyName = "x")]
    public double X { get; set; } = 0.0;
    [JsonProperty(PropertyName = "y")]
    public double Y { get; set; } = 0.0;
    [JsonProperty(PropertyName = "z")]
    public double Z { get; set; } = 0.0;
}
```

```
class Line
{
    [JsonProperty(PropertyName = "start")]
    public Point Start { get; set; }

    [JsonProperty(PropertyName = "end")]
    public Point End { get; set; }
}
```


SketchIt (appbundle)

```
class SketchItParams
```

```
{
```

```
[JsonProperty(PropertyName = "walls")]
```

```
public IList<Line> Walls { get; set; }
```

```
[JsonProperty(PropertyName = "floors")]
```

```
public IList<Ilist<Point>> Floors { get; set; }
```

```
static public SketchItParams Parse()
```

```
{
```

```
    string jsonContents = File.ReadAllText("SketchItInput.json");
```

```
    return JsonConvert.DeserializeObject<SketchItParams>(jsonContents);
```

```
}
```

```
}
```

SketchIt (appbundle)

```
static void SketchItFunc(DesignAutomationData data)
{
    Application rvtApp = data.RevitApp;
    Document newDoc = data.RevitApp.NewProjectDocument(UnitSystem.Imperial);

    SketchItParams sketchItParams = SketchItParams.Parse();

    CreateWalls(sketchItParams, newDoc);
    CreateFloors(sketchItParams, newDoc);

    newDoc.SaveAs("result.rvt");
}
```


SketchIt (appbundle)

```
static void CreateWalls(SketchItParams sketchItParams, Document newDoc)
{
    FilteredElementCollector levelCollector = new FilteredElementCollector(newDoc);
    ElementId levelId = levelCollector.OfClass(typeof(Level)).FirstElementId();

    using (Transaction transaction = new Transaction(newDoc, "Create walls"))
    {
        transaction.Start();
        foreach (var line in sketchItParams.Walls)
        {
            XYZ start = new XYZ(line.Start.X, line.Start.Y, line.Start.Z);
            XYZ end = new XYZ(line.End.X, line.End.Y, line.End.Z);
            Curve curve = Autodesk.Revit.DB.Line.CreateBound(start, end);
            Wall.Create(newDoc, curve, levelId, false);
        }
        transaction.Commit();
    }
}
```

SketchIt (workitem)

```
{  
  "activitId": "MyNickname.SketchItActivity+myalias",  
  "arguments": {  
    "sketchItInput": {  
      "url": "data:application/json, {  
        "walls": [  
          {"start": {"x": 10, "y": 10, "z": 0}, "end": {"x": 20, "y": 20, "z": 0} },  
          {"start": {"x": 30, "y": 30, "z": 0}, "end": {"x": 40, "y": 40, "z": 0} },  
        ],  
        "floors": []  
      }  
    },  
    "result": {  
      "verb": "put",  
      "url": "https://my-storage-location.com/output.json"  
    }  
  }  
}
```


SketchIt (workitem)

```
{
  "activitId": "MyNickname.SketchItActivity+myalias",
  "arguments": {
    "sketchItParams": {
      "url": "https://my-storage-location.com/large-input.json"
    },
    "result": {
      "verb": "put",
      "url": "https://my-storage-location.com/output.json"
    }
  }
}
```

SketchIt (Recap / Learnings)

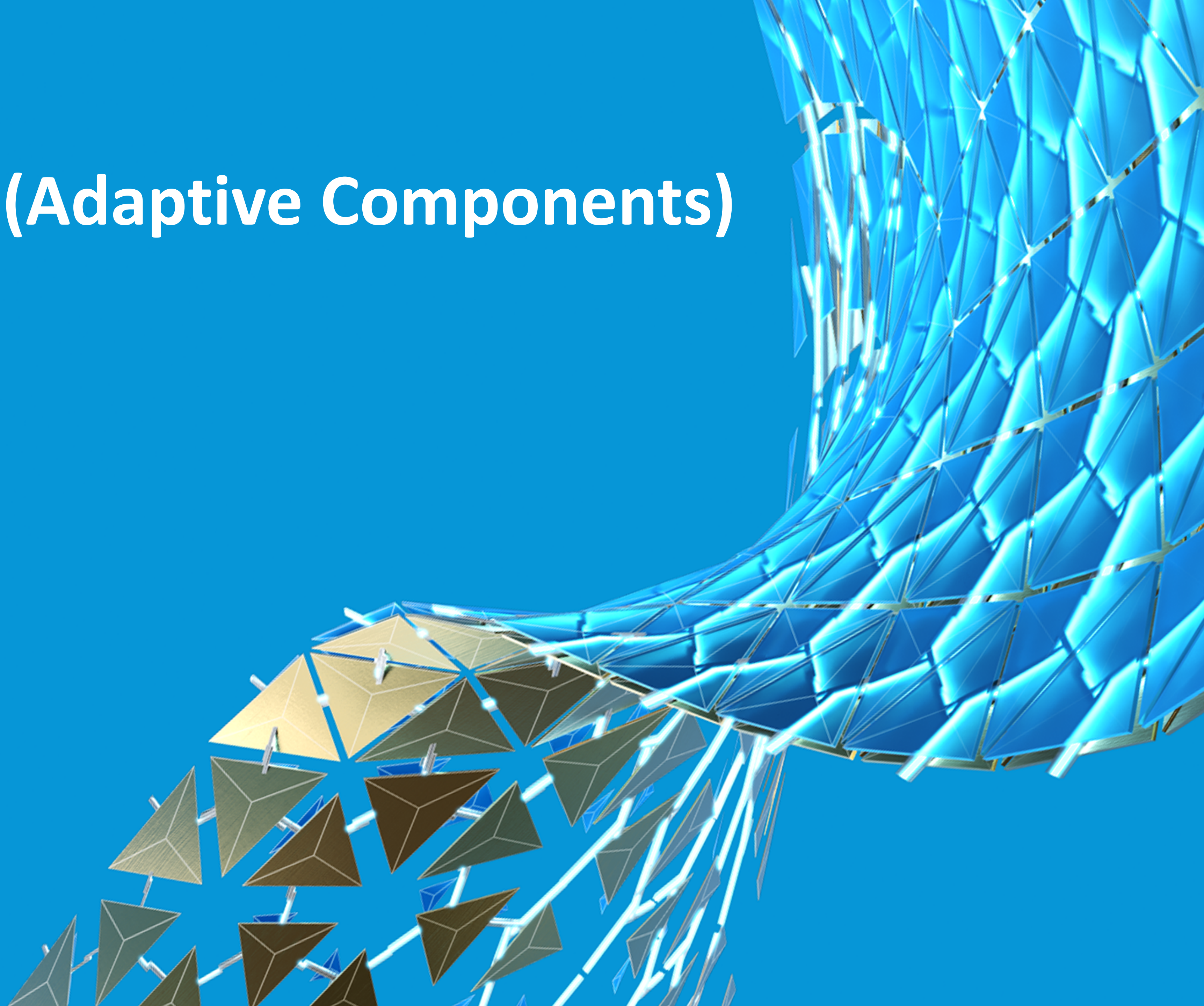
Create a new Revit document

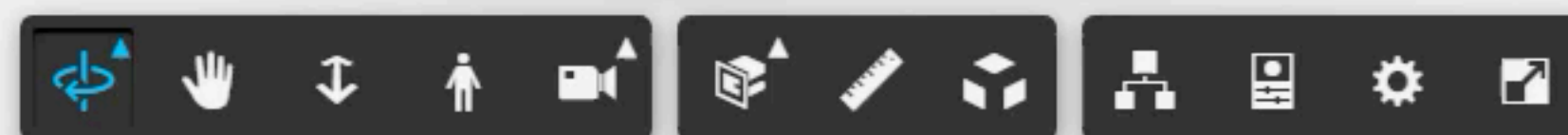
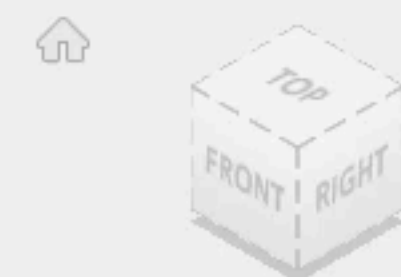
Handle composite json data

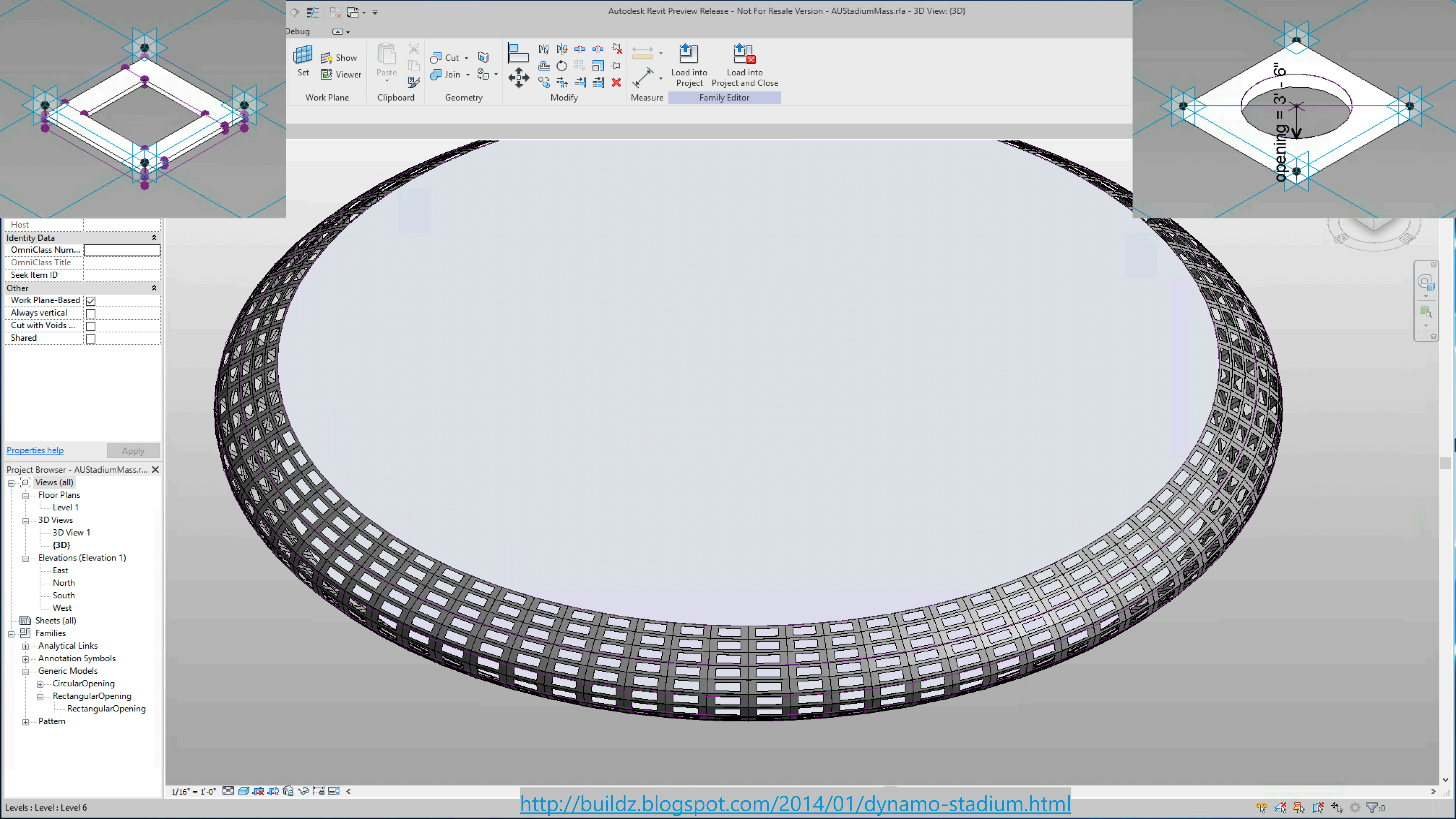
Send large json data

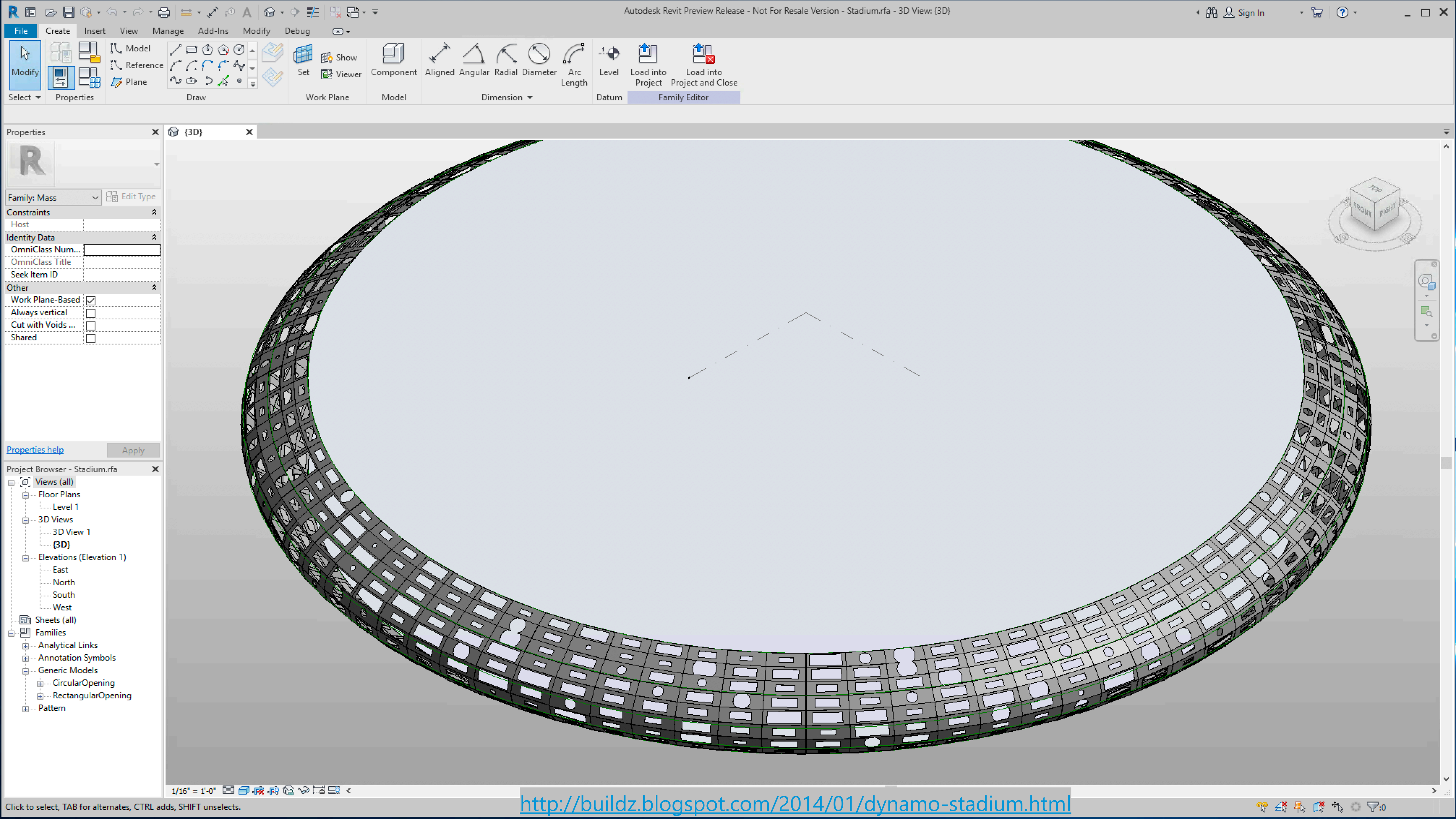
Application with creation workflow

Stadium Panels (Adaptive Components)









Stadium Panels (workitem)

```
"activitId": "MyNickname.StadiumPanelsActivity+myalias",
```

```
"arguments": {
```

```
  "rfafile": {
```

```
    "url": "https://my-storage-location.com/StadiumPanels.rfa"
```

```
  },
```

```
  "params": {
```

```
    "url": "data:application/json,{.....}"
```

```
  },
```

```
  "result": {
```

```
    "verb": "put",
```

```
    "url": "https://my-storage-location.com/StadiumProject.rvt"
```

```
  }
```

```
}
```

```
}
```


Stadium Panels (workitem/json params)

```
{  
  "data": [  
    {  
      "name": "RectangularOpening",  
      "weight": 3,  
      "params": [  
        {  
          "name": "opening",  
          "min": 0.08,  
          "max": 0.35  
        }  
      ]  
    },  
    {  
      "name": "CircularOpening",  
      "weight": 1,  
      "params": [  
        {  
          "name": "opening",  
          "min": 5,  
          "max": 25  
        }  
      ]  
    }  
  ]  
}
```

Stadium Panels (appbundle)

```
class FamilyParamsData
{
    public string Name { get; set; }
    public double Min { get; set; }
    public double Max { get; set; }
}
```

```
class FamilySymbolData
{
    public string Name { get; set; }
    public int Weight { get; set; }
    public IList<FamilyParamsData> Params { get; set; }
}
```

```
class StadiumParams
{
    public IList<FamilySymbolData> Data { get; set; }
```

```
static public StadiumParams Parse()
{
    string jsonContents = File.ReadAllText("StadiumParams.json");
    return JsonConvert.DeserializeObject<StadiumParams>(jsonContents);
}
```


Stadium Panels (appbundle)

```
void GenerateStadiumPanels(DesignAutomationData data)
{
    using (Transaction transaction = new Transaction(doc))
    {
        transaction.Start("Create Components");
        foreach (var elem in surfaces)
        {
            DividedSurface dividedSurface = elem as DividedSurface;
            dividedSurface.ChangeTypeId(rectangular);
        }

        var instances = new FilteredElementCollector(doc).OfClass(typeof(FamilyInstance)).ToElements();
        for (var i=0; i<instances.Count; i++)
        {
            var inst = instances[i] as FamilyInstance;
            RandomizeFamilyInstance(doc, inst, stadiumParams);
        }
        transaction.Commit();
    }
}
```

Stadium Panels (appbundle)

```
void RandomizeFamilyInstance(Document doc, FamilyInstance inst, StadiumParams stadiumParams)
{
    FamilySymbolData symbolData = GetWeightedRandomFamilySymbolData(stadiumParams);

    ElementId familySymbolId = GetSymbolId(doc, symbolData.Name);
    inst.ChangeTypeId(familySymbolId);

    foreach (var famParamData in symbolData.Params)
    {
        Parameter param = inst.ParametersMap.get_Item(famParamData.Name);
        double rand = GetRandomNumber(famParamData.Min, famParamData.Max);
        param.Set(rand);
    }
}
```


Stadium Panels (appbundle)

```
FamilySymbolData GetWeightedRandomFamilySymbolData(StadiumParams stadiumParams)
```

```
{
```

```
    int totalWeight = stadiumParams.Data.Aggregate(0, (acc, data) => acc + data.Weight);
```

```
    int randomInt = GetRandomInteger(totalWeight);
```

```
    int weight = 0;
```

```
    for (var i = 0; i < stadiumParams.Data.Count; i++)
```

```
    {
```

```
        weight += stadiumParams.Data[i].Weight;
```

```
        if (randomInt < weight)
```

```
            return stadiumParams.Data[i];
```

```
    }
```

```
    return null;
```

```
}
```

Stadium Panels (appbundle)

```
void SaveAsProject(DesignAutomationData data)
{
    Document newDoc = data.RevitApp.NewProjectDocument(UnitSystem.Imperial);
    Family family = data.RevitDoc.LoadFamily(newDoc);

    using (Transaction transaction = new Transaction(newDoc))
    {
        transaction.Start("Place Instance");
        var symb = newDoc.GetElement(family.GetFamilySymbolIds().Single()) as FamilySymbol;
        symbol.Activate();
        newDoc.Create.NewFamilyInstance(XYZ.Zero, symbol, StructuralType.UnknownFraming);
        transaction.Commit();
    }

    newDoc.SaveAs("StadiumProject.rvt");
}
```

Stadium Panels (Recap / Learnings)

Random selection of family type for instances

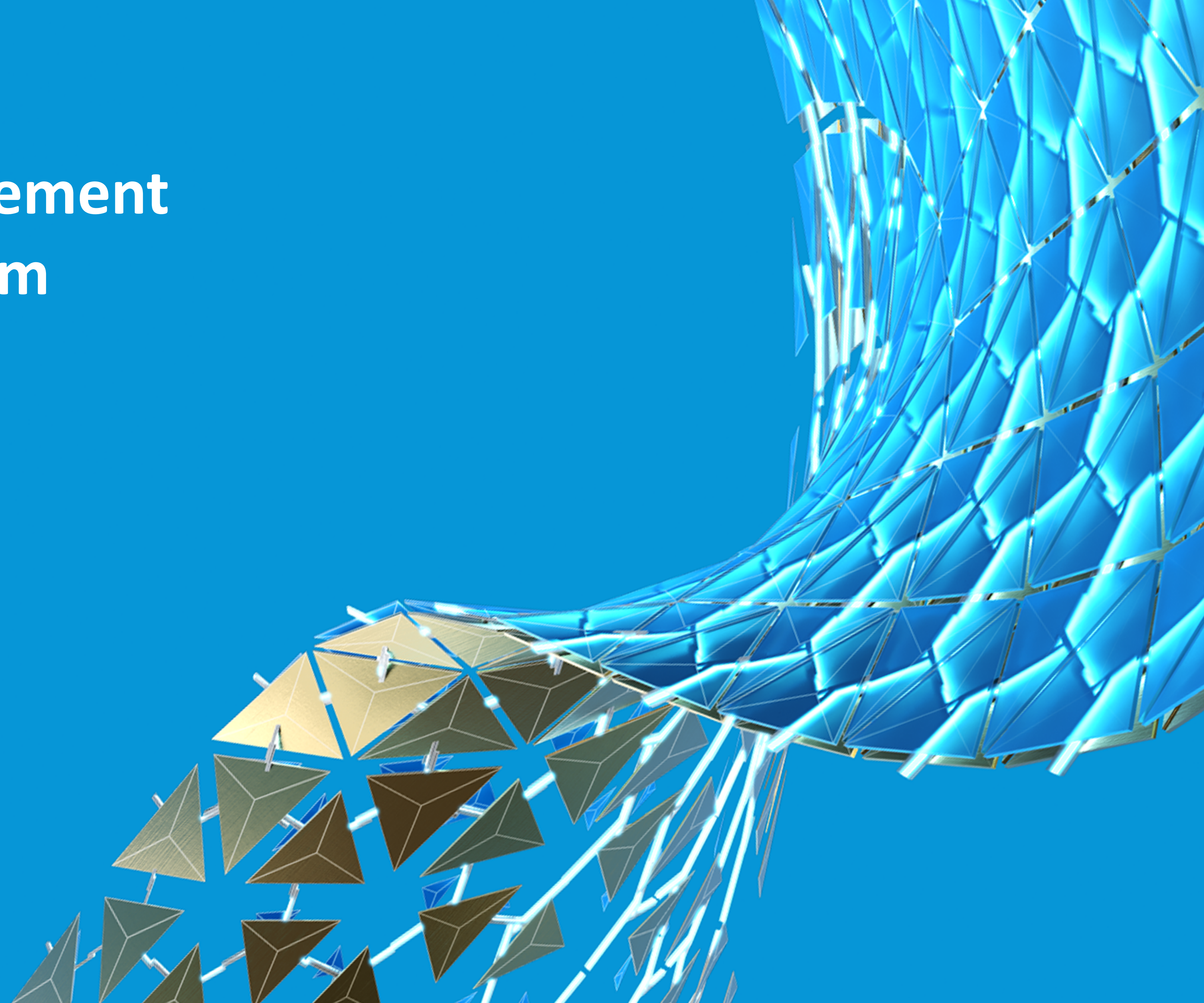
Randomly vary family parameter of instances

Workaround to view family (rfa) in Forge Viewer

Reuse families to generate flexible geometry

Power of Clould Computing to automate tedious tasks

Grid Object Placement Genetic Algorithm



Revit 2021 Generative Design

Input Parameters

Define Study

Stepped Grid Object Placement

Study Name: Stepped Grid Object Placement 007

Method: Optimize

Select in model

Select room (Element : 383205) ☒ Select

Select family instance (Element : 401268) ☒ Select

Choose variables

☒ Distance between objects - position X (m) 1 to 3

☒ Distance between objects - position Y (m) 1 to 3

☐ Minimum distance from objects to wall (m)

Set goals

☒ Number of objects ☐ Minimize ☒ Maximize

☒ Object area - overlap ☒ Minimize ☐ Maximize

☒ Object area - outside room ☒ Minimize ☐ Maximize

☒ Object area - inside room ☐ Minimize ☒ Maximize

☒ Adjustment of Y distance (%) ☒ Minimize ☐ Maximize

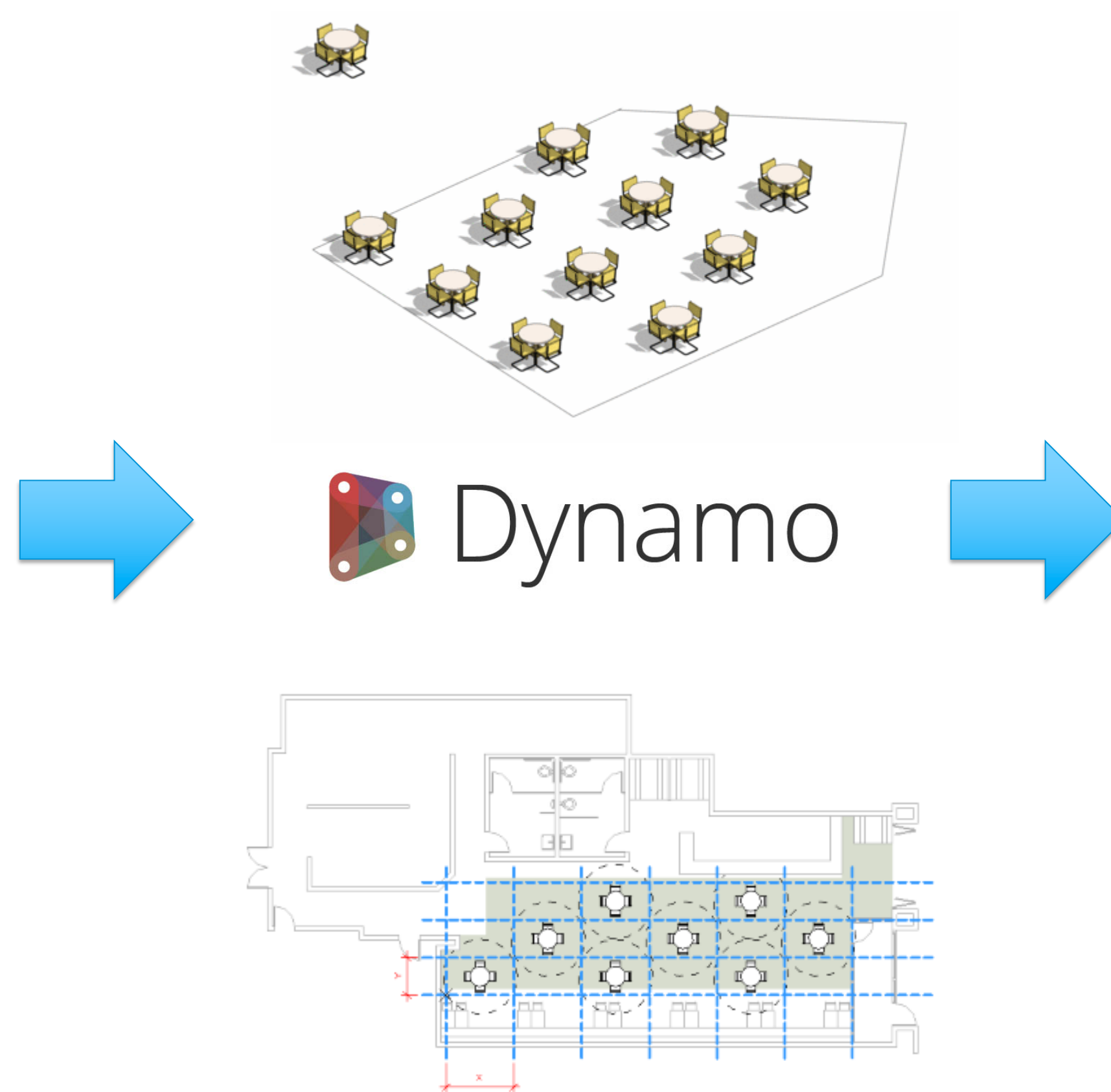
☒ Adjustment of X distance (%) ☒ Minimize ☐ Maximize

Set constraints

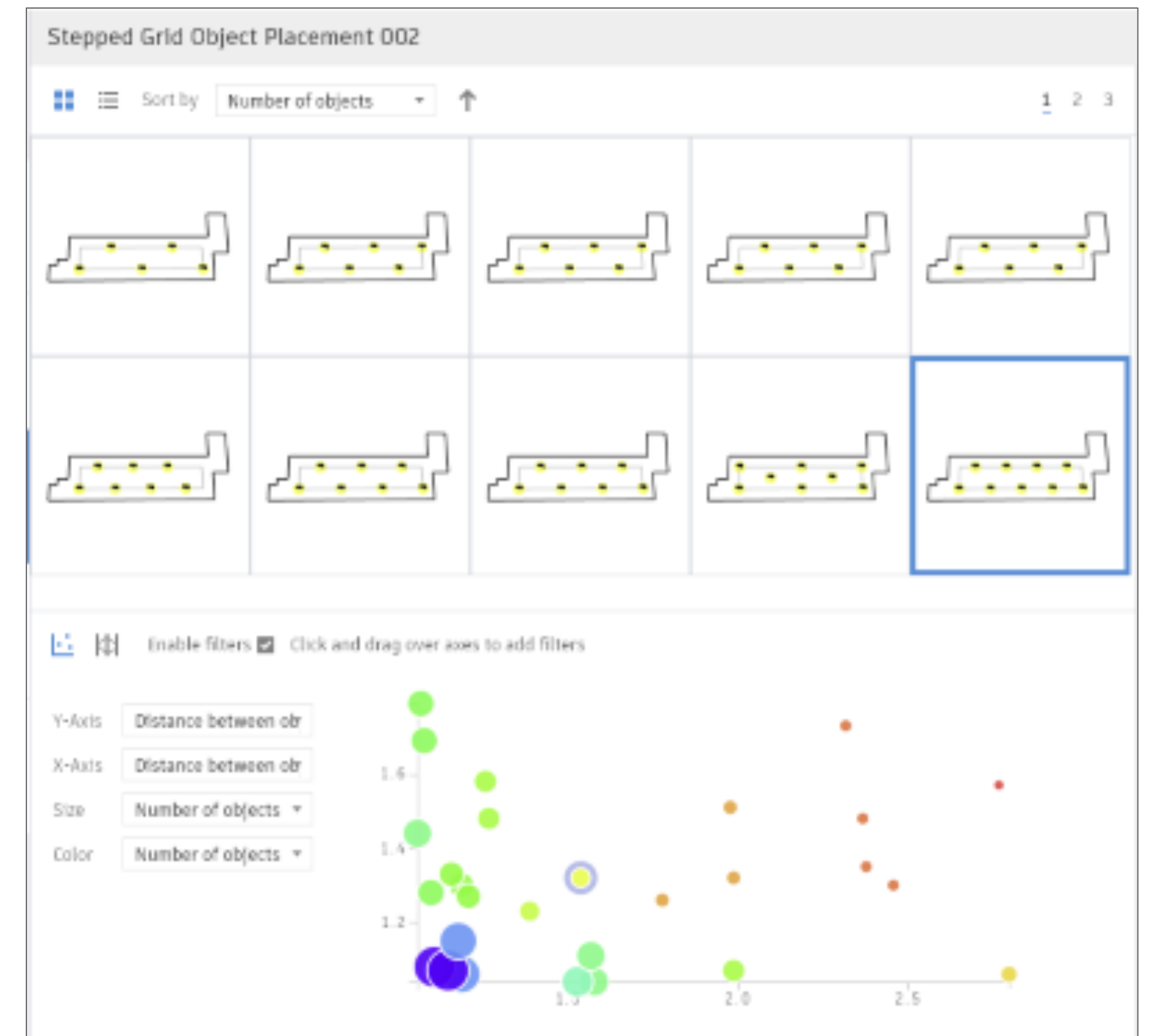
☐ Number of objects Min Max

How do I define a study?

Generating design solutions



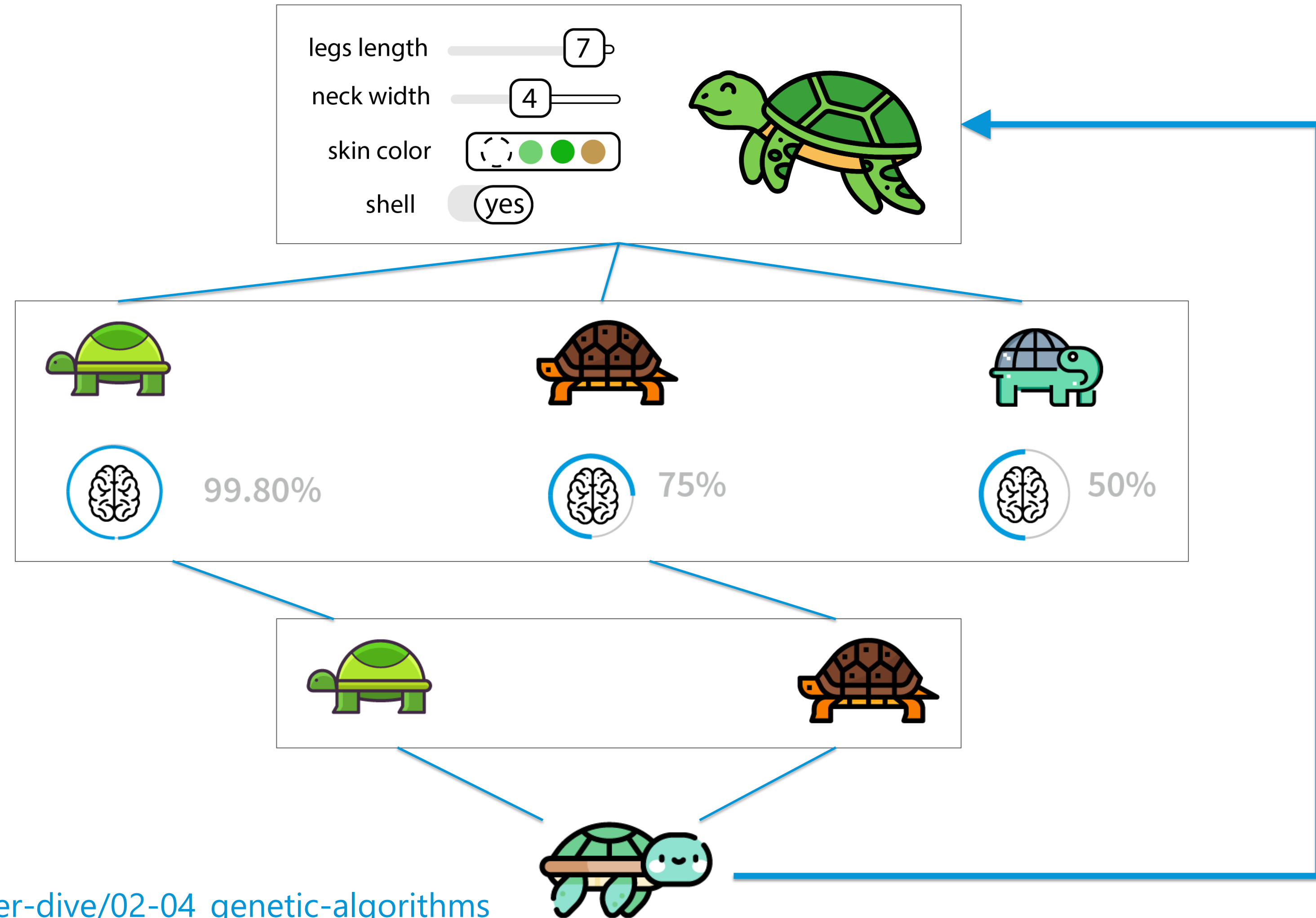
Evaluate design solutions

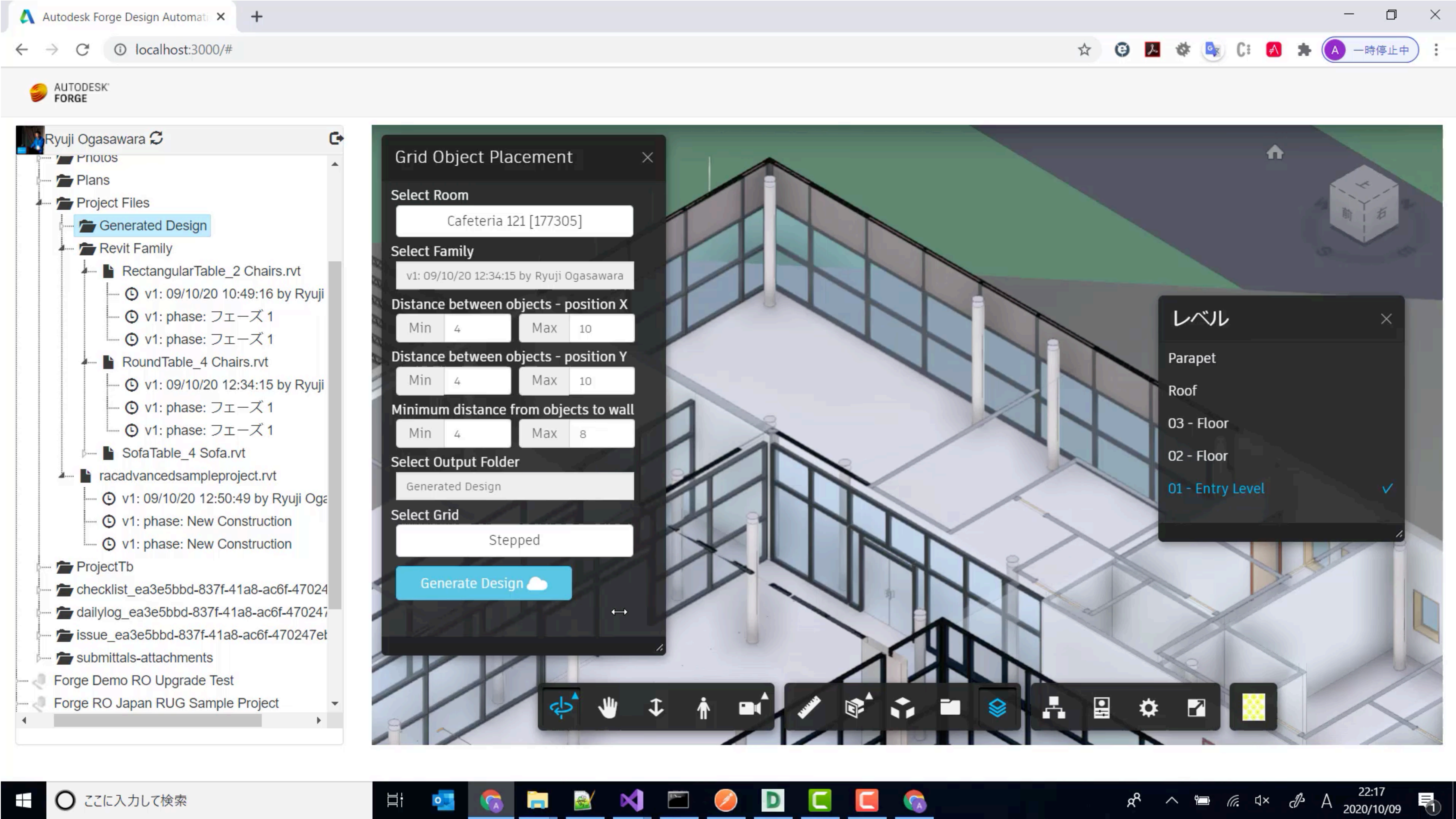


Genetic Algorithm

- **Genetic Algorithm 5 phases**

1. Initialization
2. Evaluation
3. Selection
4. Crossover / Reproduction
5. Mutation





Sample UI for input parameters

Grid Object Placement

Select Room

Room List ▼

Select Family

Select Family Project on Tree View

Distance between objects - position X

Min 4 Max 10

Distance between objects - position Y

Min 4 Max 10

Minimum distance from objects to wall

Min 4 Max 8

Select Output Folder

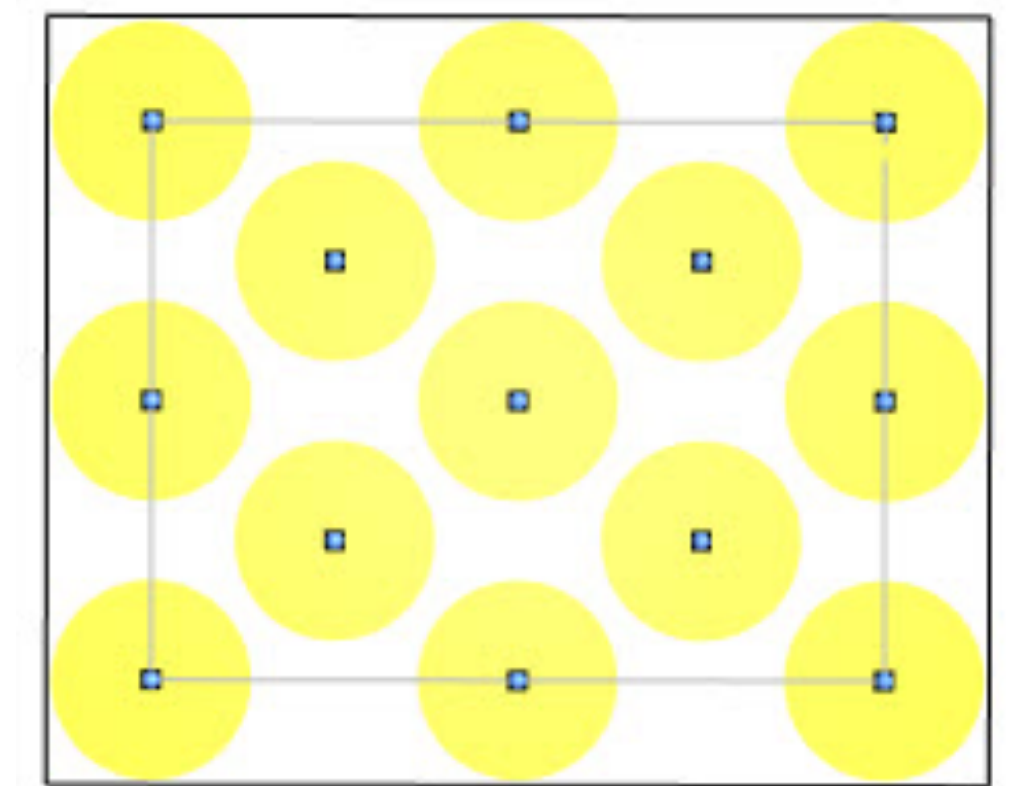
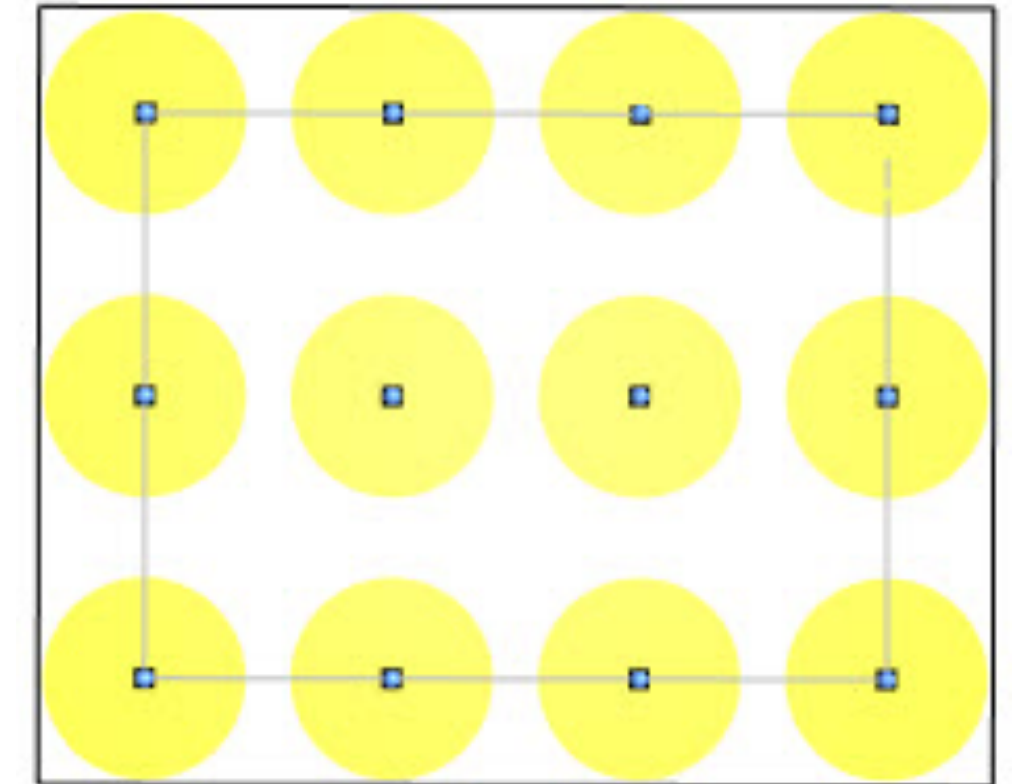
Select Output Folder on Tree View

Select Grid

Grid Type ▼

Generate Design ☁

- **Input**
 - Select a Room
 - Select a Family
 - Input Variables
 - Distance in between objects in the grid's X-axis
 - Distance in between objects in the grid's Y-axis
 - Minimum distance from objects to wall
 - Select a grid pattern
- **Output**
 - Select an output folder



Grid Object Placement (activity / parameters)

```
{
  "id": "GridObjectPlacementActivity",
  "commandLine": [ '$(engine.path)\\revitcoreconsole.exe /i "$(args[rvtFile].path)" /a1 "$(appbundles[GridObjectPlacementBundle].path)" ' ],
  "parameters": {
    "rvtFile": {
      "verb": "get",
    },
    "inputFamilyFile": {
      "verb": "get",
      "localName": "family.rvt"
    },
    "inputJsonFile": {
      "verb": "get",
      "localName": "params.json"
    },
    "resultZipFile": {
      "verb": "put",
      "zip": true,
      "localName": "exportedDwgs"
    }
  },
  "engine": "Autodesk.Revit+2021",
  "appbundles": [ "MyNickname.GridObjectPlacementBundle+myalias" ],
  "description": "."
}
```


Grid Object Placement (appbundle)

```
var chromosome = new FloatingPointChromosome(  
    new double[] { DistanceXMinParam, DistanceYMinParam, DistanceWallMinParam },  
    new double[] { DistanceXMaxParam, DistanceYMaxParam, DistanceWallMaxParam },  
    new int[] { 32, 32, 32 },  
    new int[] { 2, 2, 2 });
```

```
var population = new Population(8, 12, chromosome);
```

```
var fitness = new FuncFitness((c) => {  
    var fc = c as FloatingPointChromosome;  
  
    var values = fc.ToFloatingPoints();  
  
    m_objectDistanceX = values[0];  
    m_objectDistanceY = values[1];  
    m_minimumDistanceFromObjectToWall = values[2];
```

```
    List<XYZ> objectPlacementPointList = CreateObjectPlacementPointList();
```

```
    return objectPlacementPointList.Count;
```

```
});
```

<https://github.com/giacomelli/GeneticSharp>

The MIT License (MIT) Copyright (c) 2013 Diego Giacomelli

Grid Object Placement (appbundle)

```
var selection = new EliteSelection();  
var crossover = new UniformCrossover(0.5f);  
var mutation = new FlipBitMutation();
```

```
var termination = new OrTermination(  
    new GenerationNumberTermination(20),  
    new TimeEvolvingTermination(TimeSpan.FromMinutes(10)));
```

```
var ga = new GeneticAlgorithm(  
    population,  
    fitness,  
    selection,  
    crossover,  
    mutation);
```



```
ga.GenerationRan += (sender, e) =>
{
    var bestChromosome = ga.BestChromosome as FloatingPointChromosome;
    var bestFitness = bestChromosome.Fitness.Value;
```

```
    if (bestFitness != latestFitness)
    {
        latestFitness = bestFitness;
        phenotype = bestChromosome.ToFloatingPoints();
```

```
        m_objectDistanceX = phenotype[0];
        m_objectDistanceY = phenotype[1];
        m_minimumDistanceFromObjectToWall = phenotype[2];
```

```
        List<XYZ> objectPlacementPointList = CreateObjectPlacementPointList();
```

```
        using (Transaction tx = new Transaction(m_doc))
        {
```

```
            tx.Start("Create Family Instances");
```

```
            foreach (XYZ point in objectPlacementPointList)
```

```
            {
```

```
                m_doc.Create.NewFamilyInstance(point, m_familySymbol, m_level, NonStructural);
```

```
            }
```

```
            m_doc.Export("exportedDwgs", filename, views, new DWGExportOptions());
```

```
            tx.Rollback();
```

```
        }
```

```
    }
```

```
};
```

Grid Object Placement (Recap / Learnings)

Basics of Genetic Algorithm

Use Genetic Sharp library

Package multiple output files into a single zip file

Create temporary transaction to take a DWG snapshot

Power of Genetic Algorithm to create Generative Design



Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product and services offerings, and specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2020 Autodesk. All rights reserved.

