Introduction to Dynamo Workshop, Day 1

Last Updated: 2015/08/04

This curriculum is intended for audiences with zero to moderate experience with Dynamo as a tool for computational design in the context of Revit. Examples used are intended to introduce the fundamentals of the Dynamo platform, introduce students to a survey of useful nodes from several libraries, and evoke ideas with some common use cases. Reuse any and all of this content.

Summary Outline

- Introduction and context
    - What is Dynamo
    - How is Dynamo used in practice
    - Dynamo/Revit relationship
- Structural framing
    - Working with lists
    - Custom nodes
    - Referencing and placing Revit elements
- Adaptive components
    - Place adaptive components
    - Algorithm development
    - List mapping
- Working with tabular data
    - Read data from Excel and CSV
    - Techniques and tools for advanced data management
    - Placing multiple types of Revit families
- Geometry and Code Blocks
    - Short cuts and extra functionality in code blocks
    - Working with image data
    - Geometry concepts: parameters and normal vectors
- Auditorium seating
    - Placing and orienting Revit families
    - Geometry concepts: parameters and tangent vectors
    - Setting Revit element parameters
    - Custom analysis and visualization

Required Software

- Revit 2015 or 2016
- Dynamo 0.8.1 or later
- *DynamoTutorial* package from the Dynamo Package Manager
- Microsoft Excel 2010 or later (optional)

00 Introduction and Context

9:00 AM – 9:30 AM

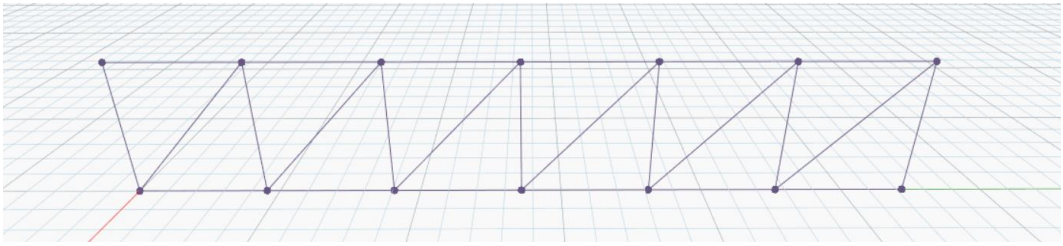
Learn what Dynamo is, how it is used in practice, and how it relates to the Revit environment.

- Presentation
    - Define computational design. Model relationships, not geometry.
    - Dynamo is for data, not just geometry.
    - Package Manager / extensions for Dynamo
    - Examples of Dynamo in practice
    - Dynamo community: forum, gallery, @DynamoBIM, Facebook, LinkedIn
- Introduction to the interface
- Dynamo/Revit interaction
    - Open Dynamo from a blank conceptual mass family in Revit
    - Make a point with **Point.ByCoordinates**
    - Make a reference point with **ReferencePoint.ByPoint**
    - Connect a slider to the input of the point while running automatically to see the reference point update. Explore effects of hosting behavior in Revit.
    - Try running manually and automatically

01 Structural Framing

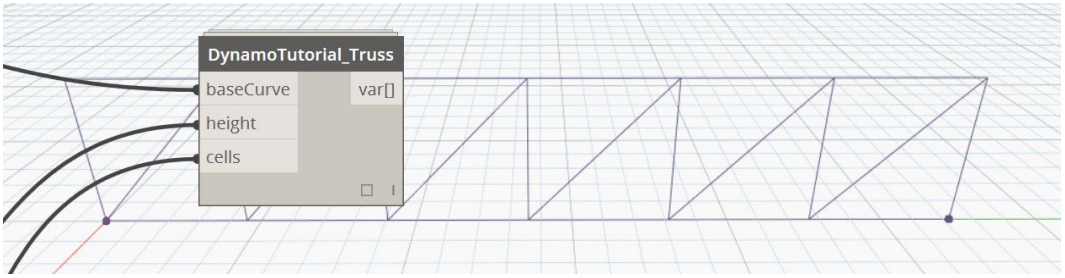
9:30 AM – 11:00 AM

Learn to work with series and basic geometry. Define a repeatable process, and apply the logic to data from a Revit project.

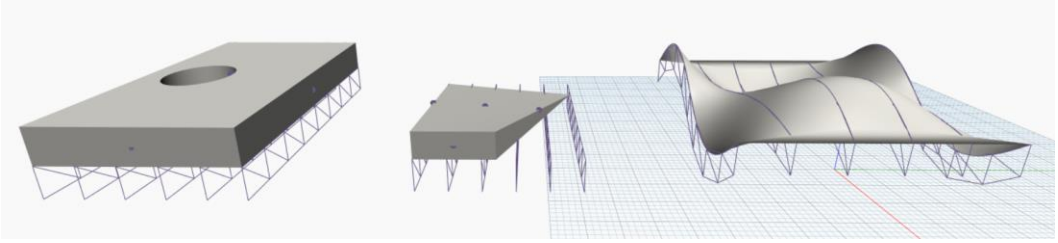


- Make a truss *[00 Truss – Simple.dyn]*
    - Set up
        - Open Dynamo from a blank architecture template in Revit
        - Download DynamoTutorial from the Package Manager
        - Draw a model curve in the Revit file
        - Select the curve from Dynamo with **Select Model Element**. Note that you will need to move the Dynamo window out of the way to see the Revit element.
        - Extract the curve from the element with **CurveElement.Curve**
    - Use range syntax to find a series of evenly-spaced points along the line with **Curve.PointAtParameter** as **0..1..#(numCells+1)**
    - Create another set of points above the first set with **Geometry.Translate**

- Side bar: demonstrate list lacing with lines between series of points *[01 List Lacing.dyn]*
- Connect the "nodes" of the truss with lines, making sure that there are no polycurves (Revit doesn't like polycurves for placing structural framing)
- Create structural framing elements with **StructuralFraming.BeamByCurve**



- Create a custom node *[02 Truss – Custom Node.dyn]*
  - Create a custom node from the truss definition leaving the curve as the only input
  - Introduce other parameters like *number of cells* and *height*.
  - Introduce data types and default values in the parameter inputs
  - Apply to several curves
  - This custom node can also be found in the DynamoTutorial library as **DynamoTutorial_Truss**



- Apply to geometry from Revit *[03 Truss – Apply to Surfaces.dyn]*
  - Open *RoofSurfaces.rvt*
  - Use **Select Face** and **Select Model Element** followed by **Element.Geometry** to reference geometry from the .rvt file.
  - In the case of multiple surfaces, find the lowest surface
  - Find isolines on the surface with **Surface.GetIsoline**.
  - Use these as the base curves for trusses and note the behavior or parameterized surfaces. Discuss other methods for finding base curves for the trusses

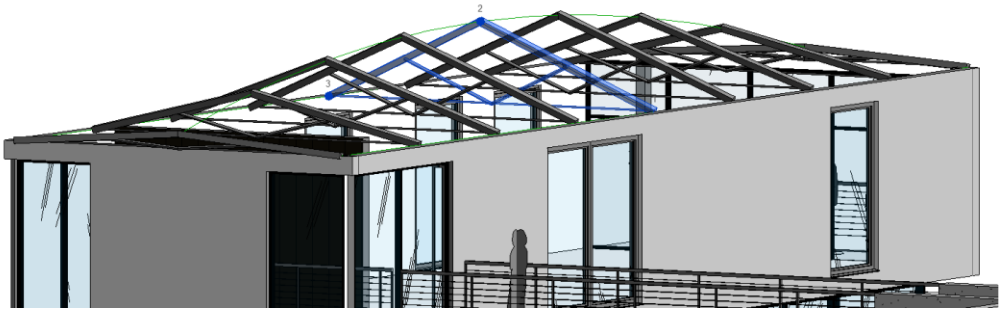
Morning Break

11:00 AM – 11:15 AM

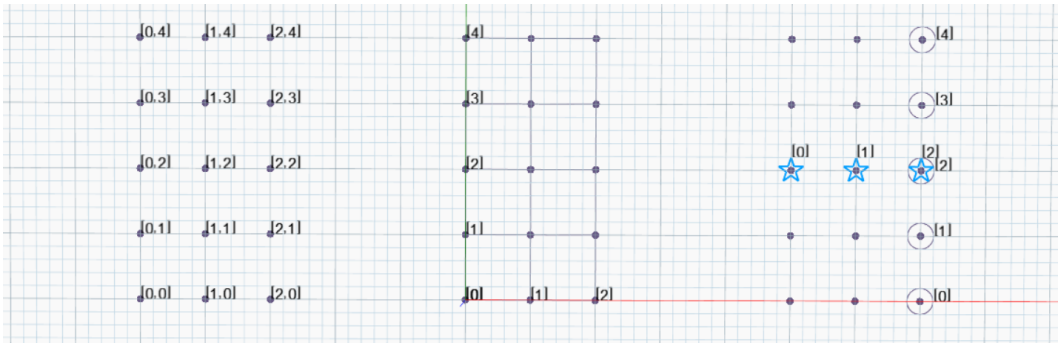
02 Adaptive Components

11:15 AM – 12:30 PM

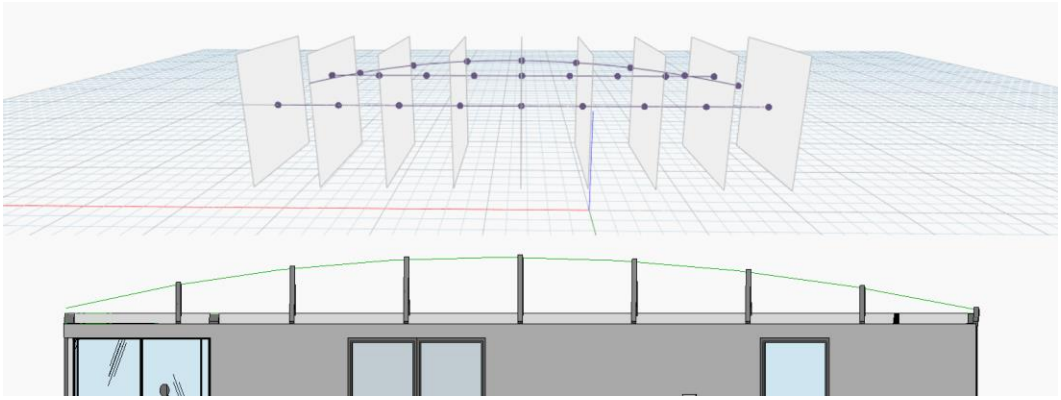
Learn how to use Dynamo to control the placement of Adaptive Components in Revit. Start with simple logic, then develop a smarter script to control geometry placement. Learn advanced data management techniques along the way.



- Use curve parameters to place adaptive component trusses *[00 AC Truss – By Parameters.dyn]*
  - Set up
    - Open a new Dynamo definition on top of *DynamoSample_2015.rvt*
    - In a new Dynamo file, select the three model curves spanning the open roof of the house. One is in the middle, and two are along the top inside edges of the walls.
  - Group the three model curves into a single list with **List.Create** to treat all at once
  - Extract the geometry from the list of model curves with **CurveElement.Curve**
  - Find a series of points along each line with range syntax, introducing a parameter for the number of trusses as **0..1..#numTrusses**, with **Curve.PointAtParameter**. Be careful to use cross product lacing.
  - Reorient the organization of the list structure with **List.Transpose** to create lists of 3 points each.
  - Place adaptive component trusses with **AdaptiveComponent.ByPoints**. The three families available in the file are:
    - Generic Models / 3PointAC
    - Generic Models / 3PointAC_SquareTruss
    - Generic Models / 3PointAC_wireTruss
  - To note:
    - While Dynamo is running automatically, move the model curves then change the number of trusses and the family type being placed to see the changes.
    - Try running Dynamo manually and hitting the Run button and undoing changes in the Revit project
    - The trusses are not constrained to be vertical. Next, we will add additional logic for this.



- Side Bar: List Mapping *[01 List Mapping.dyn]*
    - Set up
        - Open a new session of Dynamo Standalone
        - Create a small, non-square grid of points by passing a series into the **x** and **y** ports of **Point.ByCoordinates** and choosing cross product lacing
    - Explore **List.Transpose**
        - Pass the nested list of points to **PolyCurve.ByPoints**
        - Transpose the list of points and then pass to **PolyCurve.ByPoints**
    - Explore **List.Map**
        - Use **List.GetItemAtIndex** and draw circles around a row of points
        - Use the same node with **List.Map** and draw another shape around a column of points, having selected the *nth* point *from each* list.
    - Explore **List.Count**
        - How many items are in each list?
        - **List.Count** looks for lists as opposed to items of data.



- Use geometric constraints to place adaptive component trusses *[02 AC Truss – By Planes.dyn]*
    - Return to the same file used before, and start near the beginning to introduce new logic. Keep the **AdaptiveComponent.ByPoints** node from before so that the new logic will update the old trusses as opposed to making new ones.
    - Introduce the constraint that the trusses be vertical
        - Choose a “control curve” then evaluate a series of vertical planes along that curve

        - Find the intersections between those planes and the set of curves, and note cross product lacing on **Geometry.Intersect**.
    - Adjust the list structure with **Flatten** and **List.Map**
    - Filter out lists of points that contain anything other than 3 points
    - Place the Adaptive components.

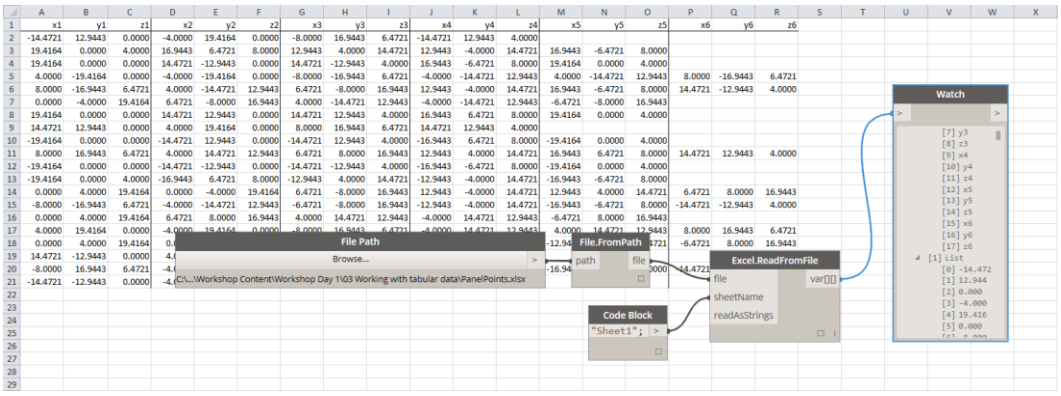
Lunch

12:30 PM – 1:30 PM

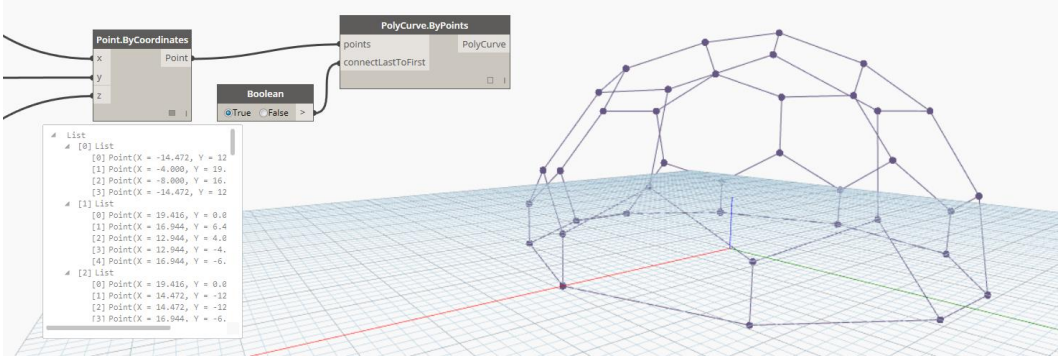
03 Working with Tabular Data

1:30 PM – 2:15 PM

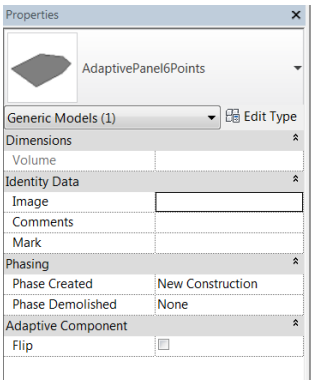
Read and manipulate data from an Excel spreadsheet to place Revit families. Use techniques and tools for advanced data management.

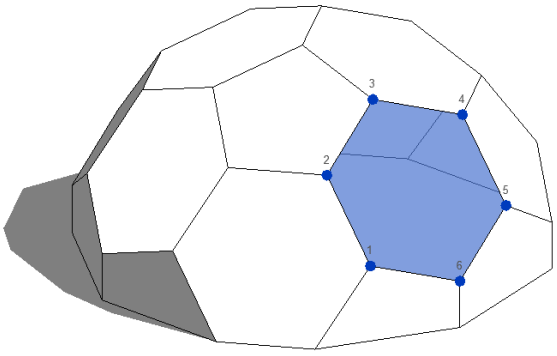


- Read Data from Excel *[00 Panels from Excel – Read Data.dyn]*
    - Set up
        - Open a new Dynamo file on top of *PanelStructure_Start.rvt*
        - Note that the file contains 3 adaptive component families:
            - Generic Models / AdaptivePanel4Points
            - Generic Models / AdaptivePanel5Points
            - Generic Models / AdaptivePanel6Points
        - Open *PanelPoints.xlsx* or *PanelPoints.csv*. The data are organized as series of x-, y-, and z-coordinates of points that describe panels. Panels may have 4, 5, or 6 sides.
    - Use **File Path** to select the Excel (or CSV) file PanelPoints. The node **File.FromPath** sets up a file watcher to make sure Dynamo notices changes to the file and updates the graph accordingly.
    - Use **Excel.ReadFromFile** or **CSV.ReadFromFile** to read the data. It appears as a nested list, with each sub list representing a row of data.



- Organize lists
    - Use the first row of data as a "sketch" to figure out how to sort the data into lists of x-, y-, and z-coordinates, with each title block representing the column of numbers.
    - **List.Clean** removes null values which result from reading empty cells in the Excel file, or which fill in missing data in a rectangular matrix in the CSV file.
    - Use **List.Map** to apply the logic used for the title row *to each* of the columns.



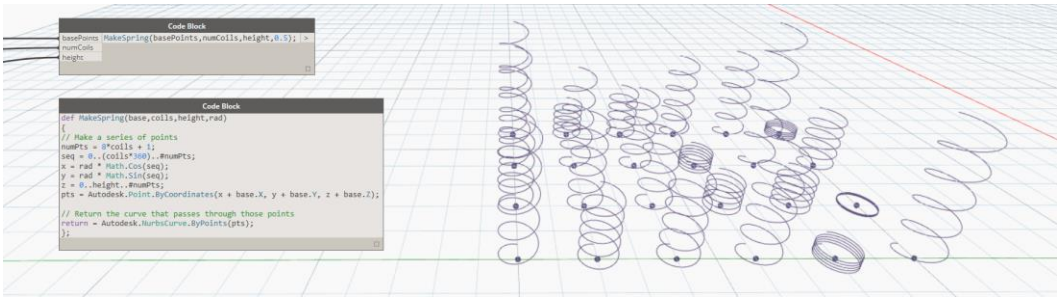


- Place families
    - Sort the lists of points into groups depending on whether there are 4, 5, or 6 points. Use **List.GroupByKey** from the Ampersand package (which downloads from the Package Manager with the DynamoTutorial package), with the "key" value as the number of points in each list.
    - Construct a list of the family types to be placed corresponding to the order of the groups. See the list of adaptive component panels listed above.
    - Use **List.Combine**, which is the 2-or-more-dimensional version of **List.Map**, to apply the list of family types to the list of groups of points.
    - With Dynamo running automatically, try making a change to the Excel or CSV files (and saving that file with the change).
- Going Further: extend Dynamo with user-created packages (see Packages menu)
    - Bumblebee for Excel interoperability
    - Lunchbox for panelization tools
    - Dynableau for interoperability with Tableau
    - Raindrops for interoperability with Google Spreadsheets

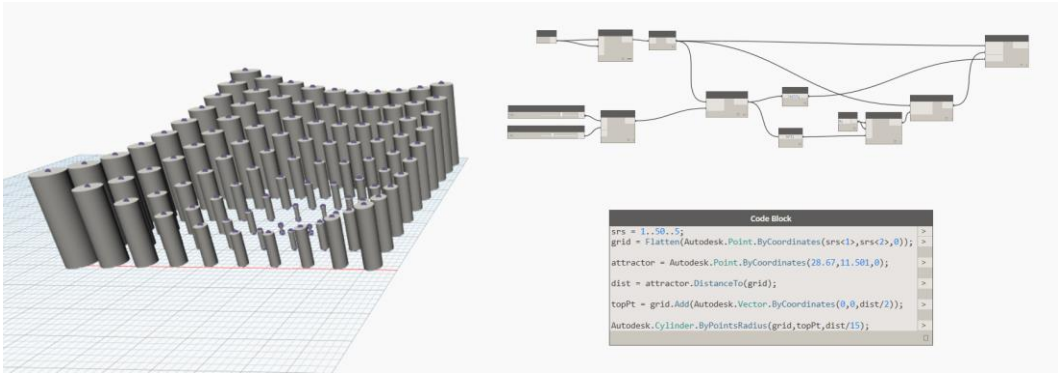
04 Geometry and Code Blocks

2:15 PM – 3:00 PM

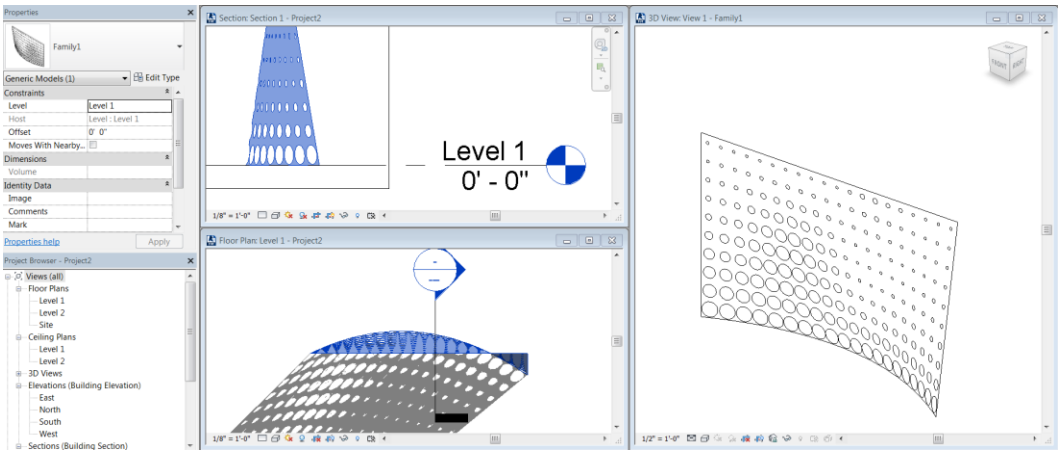
Learn how to use and understand code blocks to get more out of your script including shortcuts and options for making concise and reusable code.



- Intro to Code Blocks *[00 Intro to Code Blocks.dyn]*
    - Familiar shortcuts: range..syntax, accessing list[items], {creating lists}, “strings”
    - Calling other nodes from the code block
    - Defining functions



- Translate nodes to code *[01 Node to Code.dyn]*
    - Set up: open *01 Node to Code START.dyn*
    - Methodically, translate the nodes into a single code block
    - Can also right-click for automatic Node 2 Code in v. 0.8.2 and on later.



- Create a perforated screen *[02 Perforated Screen.dyn]*
    - Set up
        - Open a new Generic Model family in Revit. Set the units to Imperial or scale the starting surface with **Geomtery.Scale** to something appropriate to the units of choice.
        - Open *02 Perforated Screen START.dyn*
        - Navigate to *gradient.jpg* or your own image.
    - Wrangle the pixel color data into a similar format as the parameterization of the surface
    - Construct planes at each "pixel point" on the surface
    - Cut holes in the surface
        - Planar circles will not cut a curved surface, so turn the circles into cylinders which pass through the surface.
        - Trim the surface with the solid cylinders
    - Import into the Revit family, which can then be placed in the Revit project environment.
    - Export the data directly from Dynamo
        - Export to an SAT file with **ExportToSAT**.
        - Export to an STL file with File/Export Model to STL… Note that the tessellation of the STL is determined by the slider in the menu: Settings/Render Precision.

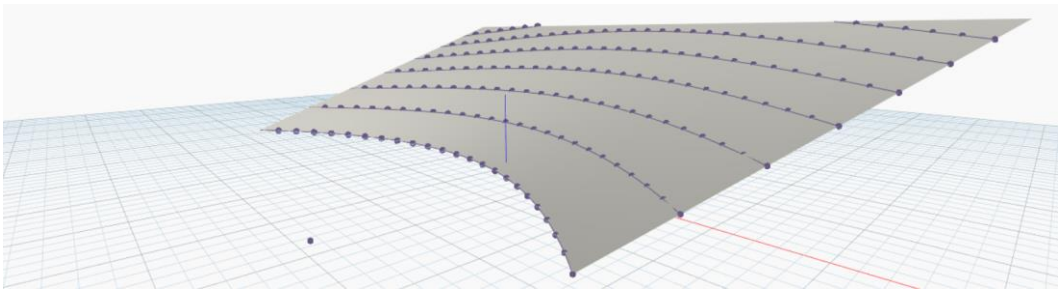
Afternoon Break

3:00 PM – 3:15 PM

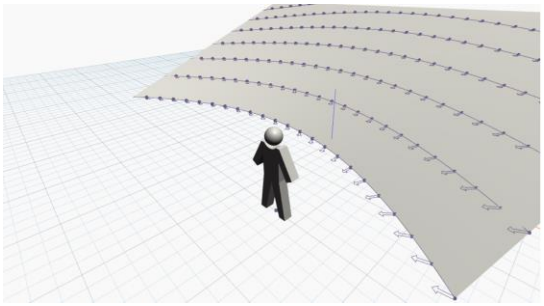
05 Auditorium Seating

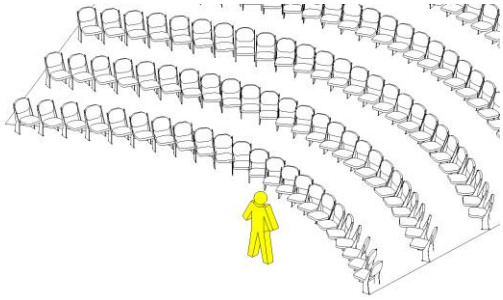
3:30 PM – 5:00 PM

Build on previous knowledge of data management, Revit families, and tabular data to place and orient seating family instances in a Revit project. Work with geometry concepts of parameters and vectors. Set Revit element parameters, and perform a custom sightline analysis.

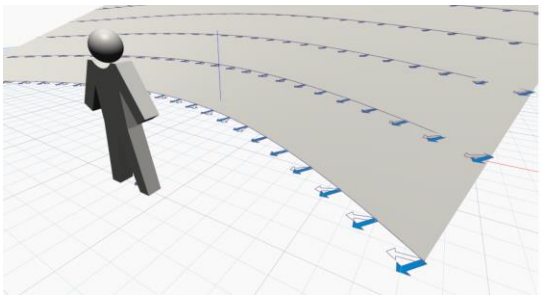


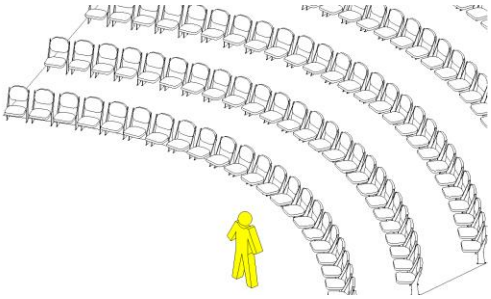
- Find seat locations *[00 Find Seat Locations.dyn]*
    - Set up
        - Open a new Dynamo file on top of *Auditorium_Start.rvt*
        - If you do not see a sloped surface, enable "Show Mass" in visibility graphics.
        - Notice the family chairPointer, which is a seating family nested inside an adaptive component to enable custom orientations.
    - Collect inputs about the auditorium geometry, the podium location, and the chair family
    - Find the base curves for the seating rows
        - Create a vertical surface extrusion from the front edge of the slope.
        - Use **Surface.Offset** and **Geometry.Intersect** to find successive row curves.
        - Use **List.Clean** and **Flatten** to get rid of empty lists and unneeded hierarchy.
    - Find the base points for the seats
        - Account for the fact that the chair is placed by the point at the back left leg.
        - Use **Curve.PointAtDistance** to space the seats by their widths.



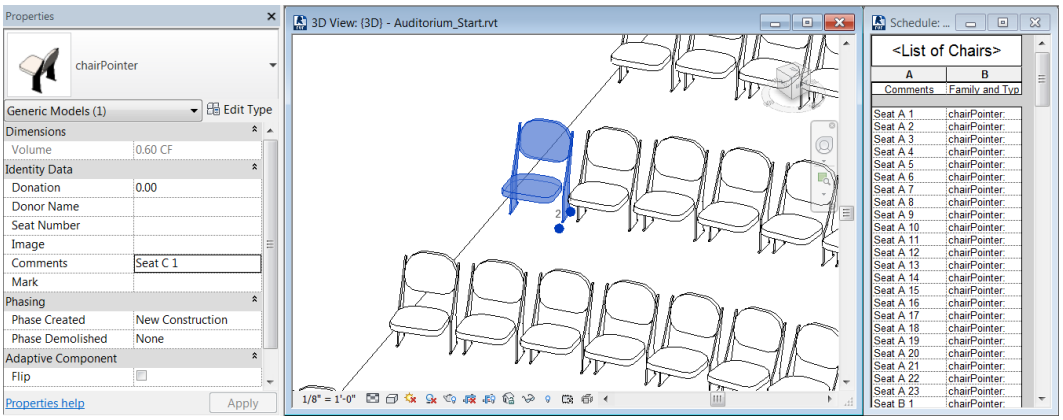


- Place chair family instances *[01 Place Seats.dyn]*
    - Find the second point to orient the chairs. For now, use a vector toward the speaker position.
        - Make sure to force the z-coordinate of the vector to be 0 so that the chairs are level.
        - Use the custom node **DynamoTutorial_TestOrientation** to visualize how the chairs will point.
    - Place seats being careful to respect the list structure when combining the two sets of points into matched pairs.

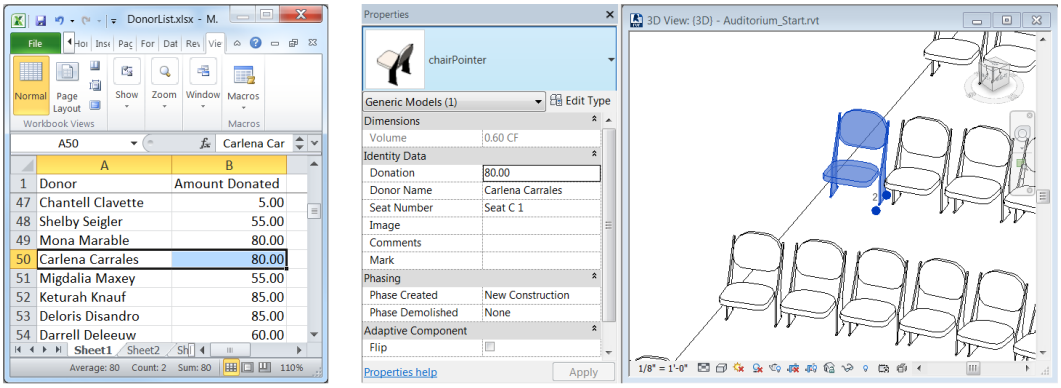




- Reorient seats *[02 Align Seats By Row.dyn]*
    - Try a different logic for orienting the seats, aligning the seats with the curve of the row instead of the position of the speaker.
    - Update the seating family instances with the new orientation points by simple substitution.

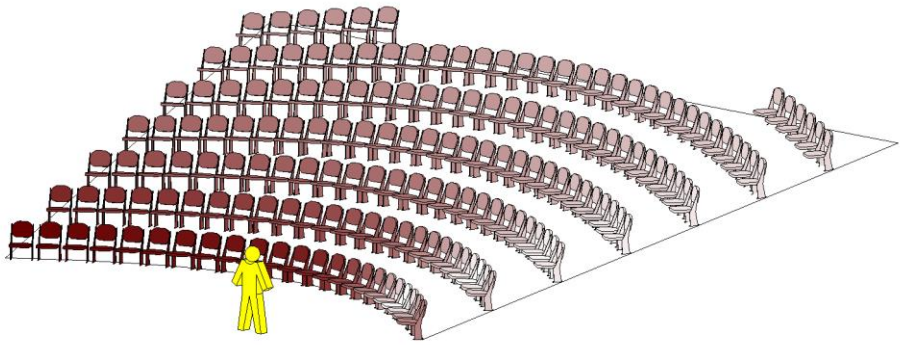


- Label seats *[03 Label Seats.dyn]*
    - Fill in the seating parameter “Seat Number” by logic for “A1, A2,… , B1, B2,…”
    - Use **Element.SetParameterByName**.



- Map data to the seats from Excel *[04 Map Data To Seats.dyn]*
    - The file *DonorNames.xlsx* (or the CSV version) contains the names of donors and donation amounts. Read these data from the file, sort the donors and donations by the size of the donation, then write the name of the donor to the “Donor Name” parameter and the amount of the donation to the “Donation” parameter for each seat in

preparation for commemorative plaques to be ordered. Donors who donated the most should have their names closes to the stage.



- Analysis *[05 Analyze Sightlines.dyn]*
  - Determine where the most uncomfortable seats in the theater are based on how far the occupant must turn their head to view the speaker's position.
  - Compare the angle between the two different orientations we used with **Vector.AngleBetween**.
  - Map these numbers to the range between 0 and 1 with **Math.RemapRange**. Use a custom **Color Range** to convert the numbers to colors.
  - Use **Element.OverrideColorInView** to color the seating instances in the Revit project.
  - While running Dynamo automatically, move the speaker!