AUTODESK UNIVERSITY

IDS22194-L

Dynamo: Generative Modeling with T-Splines

Ronnie Parsons
Mode Lab

Learning Objectives

- Create a Dynamo Graph using T-Splines Nodes
- Extend Dynamo Functionality using the Package Manager
- Export a Dynamo T-Splines file for Continued Development in an External Application
- Export a Dynamo File for Rapid Prototyping

Description

Dynamo is a visual programming platform to compose custom algorithms for processing data and generating geometry. Forward looking organizations, big and small, are leveraging this approach to problem solving to yield new and unimagined products and drive innovation. In this session, we'll explore several generative modeling techniques to generate structures, textures, and forms for rapid prototyping using T-Splines geometry. We'll look at examples of piping, surfacing, and even how to use T-Splines primitives to create Speedforms. Additionally, we'll look at some best practices for importing and exporting data from external applications. After this session, you'll come away with several useful tools to help you explore generative modeling with Dynamo and T-Splines.

Your AU Expert(s)

Ronnie Parsons is the co-founder and CEO of Mode Lab, a global design and strategy consultancy with offices in New York and Portland, OR. He has spent the last 10 years of his career investigating new ways to connect and configure client workflows by strategically aligning product vision with user-centered technology platforms. He has extensive knowledge around the Autodesk Product Design products including Fusion360, Speedform, and Dynamo. He and his team have partnered with Autodesk to provide customers with a better understanding of how generative modeling can help solve their business needs. These areas of partnership include the Dynamo Primer, Dynamo Dictionary, and Dynamo Video Training Curriculum.

rparsons@modelab.is Mode Lab

Create a Dynamo Graph using T-Splines Nodes

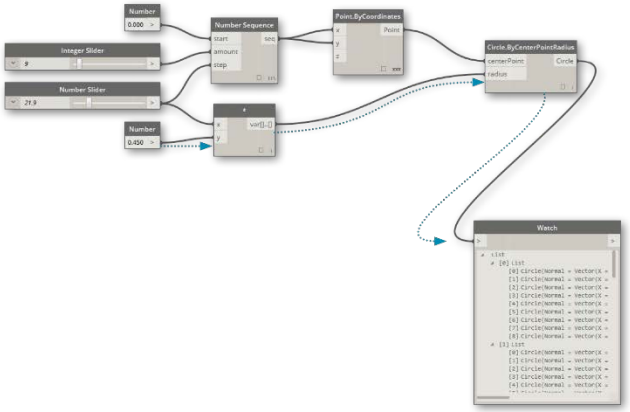
Dynamo is a visual programming language that will help you compose custom algorithms to process data and generate geometry. With Dynamo, you can generate code through a graphical (or "Visual") user interface instead of typing text bound by syntax. Creating code with Dynamo is as easy as connecting pre-packaged nodes together. Combining this powerful coding capability with the unique geometric workflows presented by T-Splines offers a robust approach to generative modeling.

What is Dynamo?

At its core, Dynamo is a platform for Visual Programming - it is a flexible and extensible design tool. Because it can operate as a stand-alone application or as an add-on to other design software, users can develop a wide range of creative workflows.

The Process

Dynamo graphs are a collection of nodes, which represent objects or functions, that are wired together to form a set of instructions on how to process data and/or build geometry. How nodes are wired together determines the order of operations. Data flows between connected nodes and is executed in the graph from left to right.



DYNAMO GRAPH

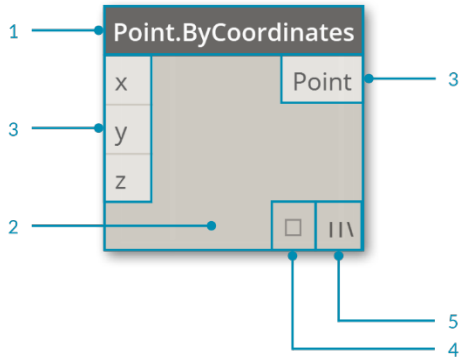
Anatomy of a Node

In Dynamo, Nodes are the objects you connect to form a Visual Program. Each Node performs an operation - sometimes that may be as simple as storing a number or it may be a more complex action such as creating or querying geometry.

Most Nodes in Dynamo are composed of five parts. While there are exceptions, such as Input Nodes, the anatomy of each Node can be described as follows:

1. Name - The Name of the Node with a Category.Name naming convention
2. Main - The main body of the Node - Right-clicking here presents options at the level of the whole Node
3. Ports (In and Out) - The receptors for Wires that supply the input data to the Node as well as the results of the Node's action
4. Data Preview - Hover or click to see a tooltip describing the results of the Node's action

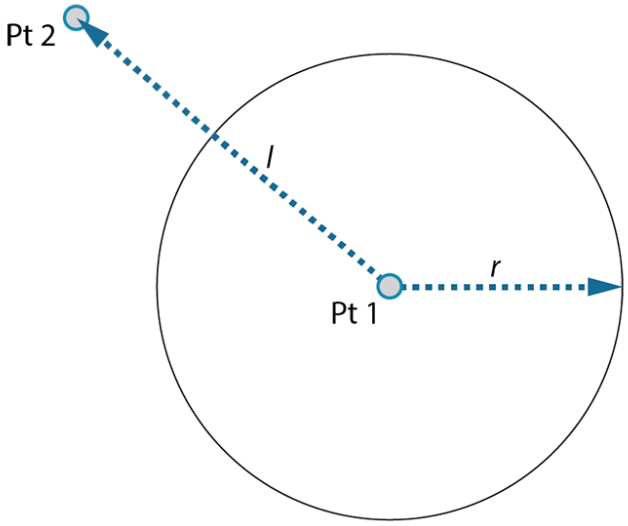
5. Lacing Icon - Indicates the Lacing option specified for matching list inputs (more on that later)



DYNAMO NODE

Defining Objectives and Relationships

Before we add anything to the Dynamo Workspace, it is key that we have a solid understanding of what we are trying to achieve and what the significant relationships will be. Remember that anytime we are connecting two Nodes, we are creating an explicit link between them - we may change the flow of data later, but once connected we've commited to that relationship. In this exercise we want to create a circle (Objective) where the radius input is defined by a distance to a nearby point (Relationship).

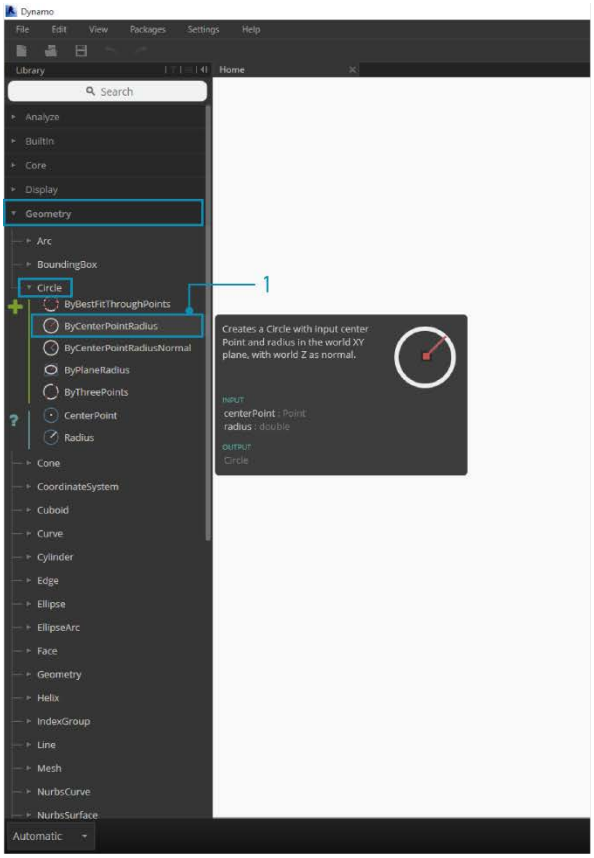


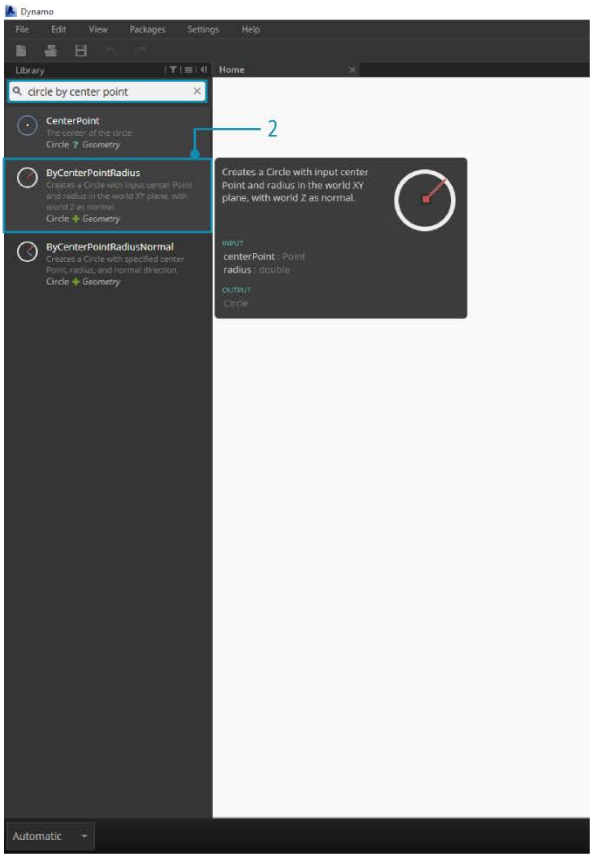
Length (*l*) ∝ Radius (*r*)

A point that defines a distance-based relationship is commonly referred to as an "Attractor." Here the distance to our Attractor Point will be used to specify how big our circle should be.

Adding Nodes to the Workspace

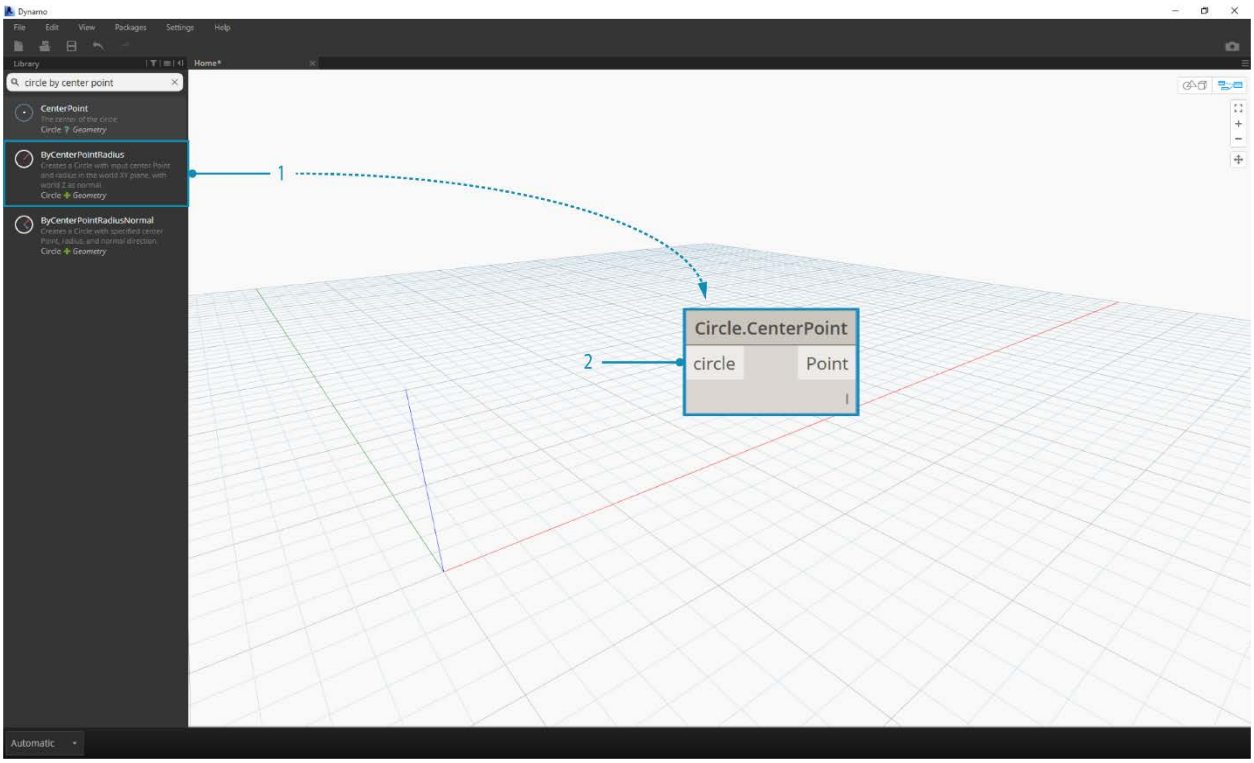
Now that we have our Objectives and Relationships sketched we can begin creating our graph. We need the Nodes that will represent the sequence of actions Dynamo will execute. Since we know we are trying to create a circle, let's start by locating a Node that does so. Using the Search field or browsing through the Library, we will find that there is more than one way to create a circle.





1. Browse to Geometry > Circle > Circle.ByPointRadius
2. Search > "Circle by Point..."

Let's add the **Circle.ByPointRadius** Node to the Workspace by clicking on it in the Library - this should add the Node to the center of the Workspace.

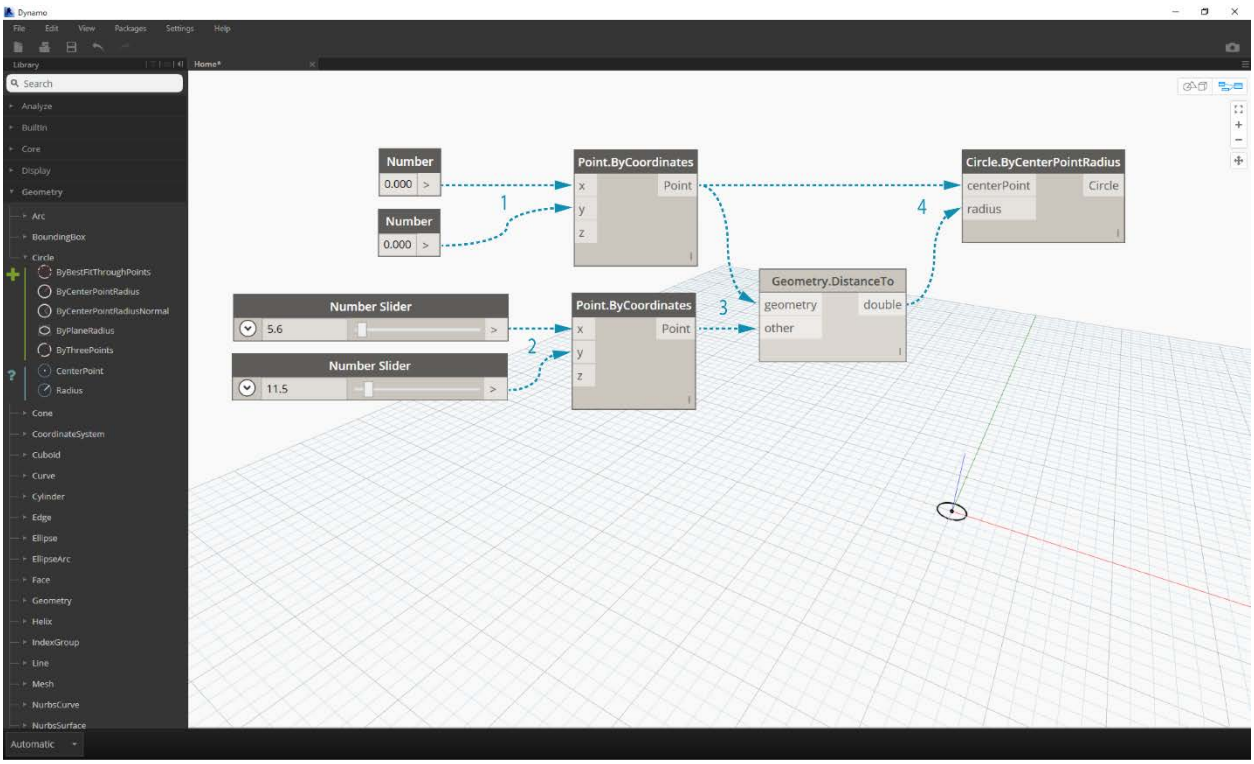


1. The **Circle.ByPointandRadius** Node in the Library
2. Clicking the Node in the Library adds it to the Workspace

We will also need **Point.ByCoordinates**, **Number Input**, and **Number Slider** Nodes.

Connecting Nodes with Wires

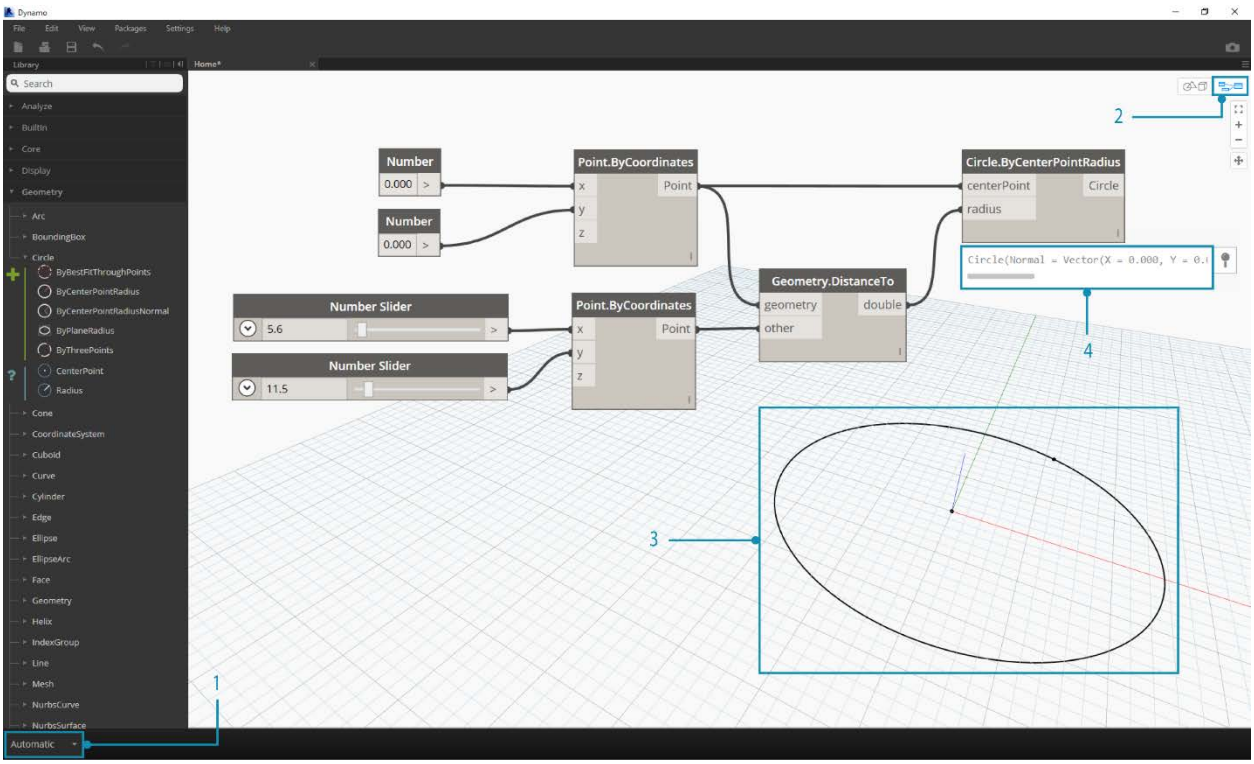
Now that we have a few Nodes, we need to connect the Ports of the Nodes with Wires. These connections will define the flow of data.



1. **Number** to **Point.ByCoordinates**
2. **Number Sliders** to **Point.ByCoordinates**
3. **Point.ByCoordinates** (2) to **DistanceTo**
4. **Point.ByCoordinates** and **DistanceTo** to **Circle.ByCenterPointRadius**

Executing the Program

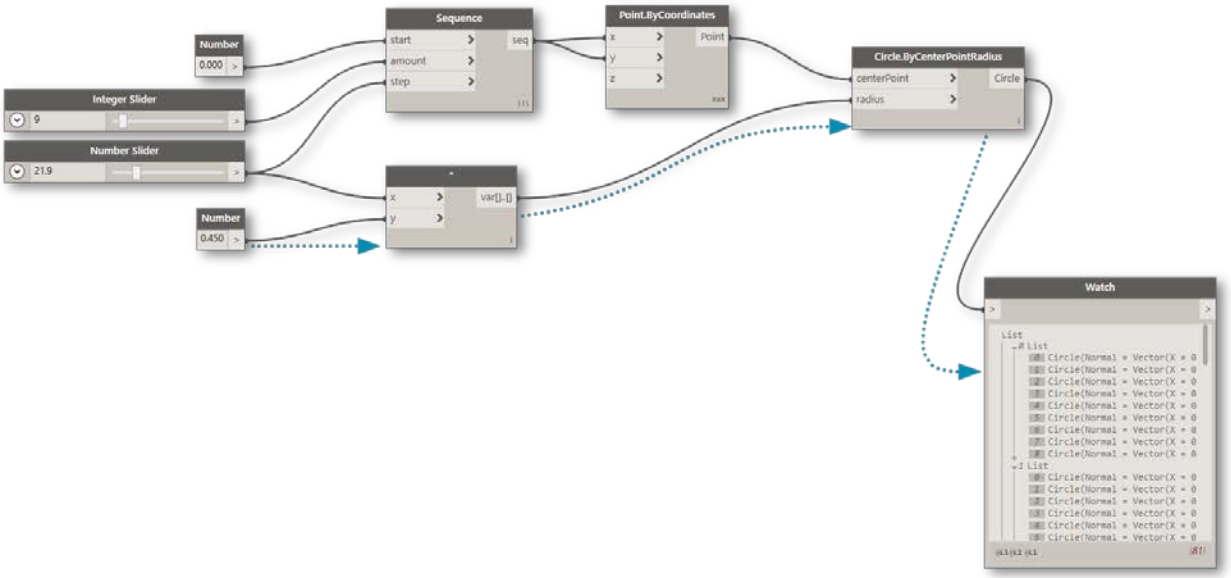
With our Program Flow defined, all we need to do is tell Dynamo to execute it. Once our program is executed (either Automatically or when we click Run in Manual Mode), data will pass through the Wires, and we should see the results in the 3d Preview.



1. (Click Run) - If the Execution Bar is in Manual Mode, we need to Click Run to execute the graph
2. Node Preview - Hovering your mouse over the box on the lower right corner of a Node will give you a pop-up of the results
3. 3D Preview - If any of our Nodes create geometry, we will see it in the 3D Preview.

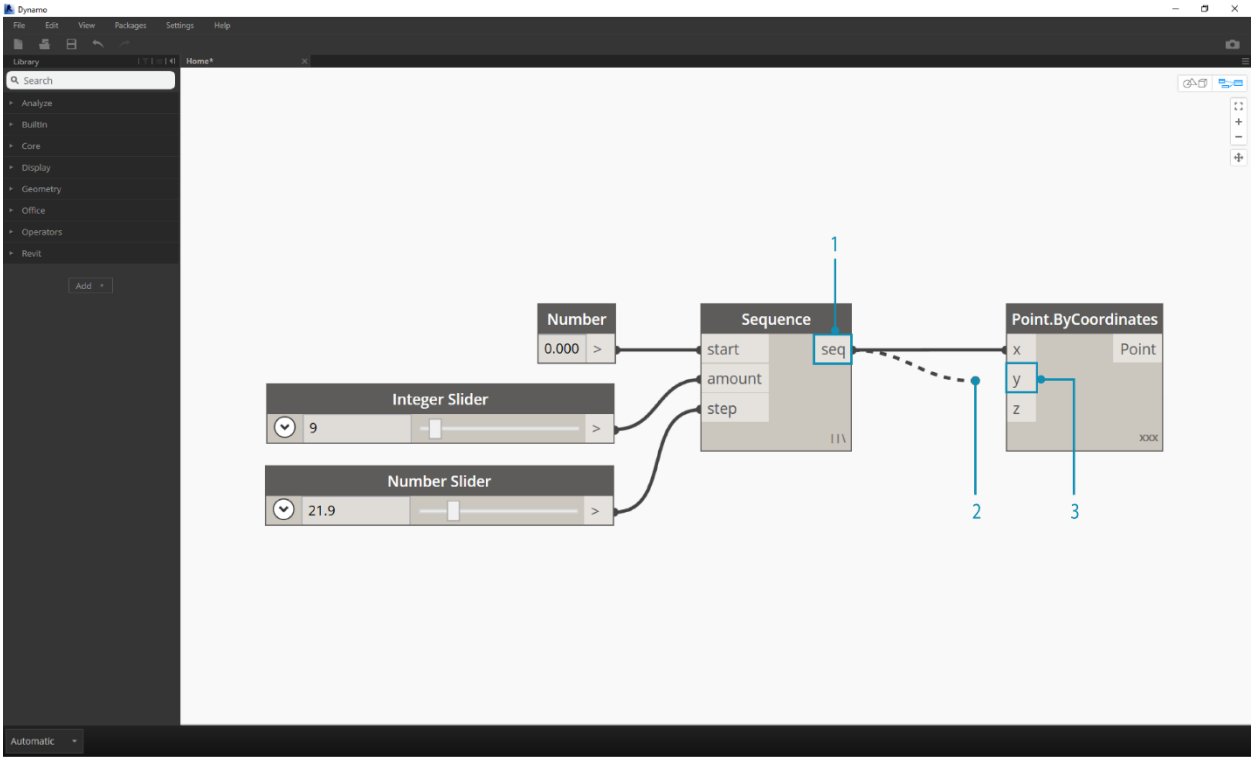
Program Flow

Wires connect the output Port from one Node to the input Port of another Node. This directionality establishes the Flow of Data in the Visual Program. Although we can arrange our Nodes however we desire in the Workspace, because the output Ports are located on the right side of Nodes and the input Ports are on the left side, we can generally say that the Program Flow moves from left to right.



Creating Wires

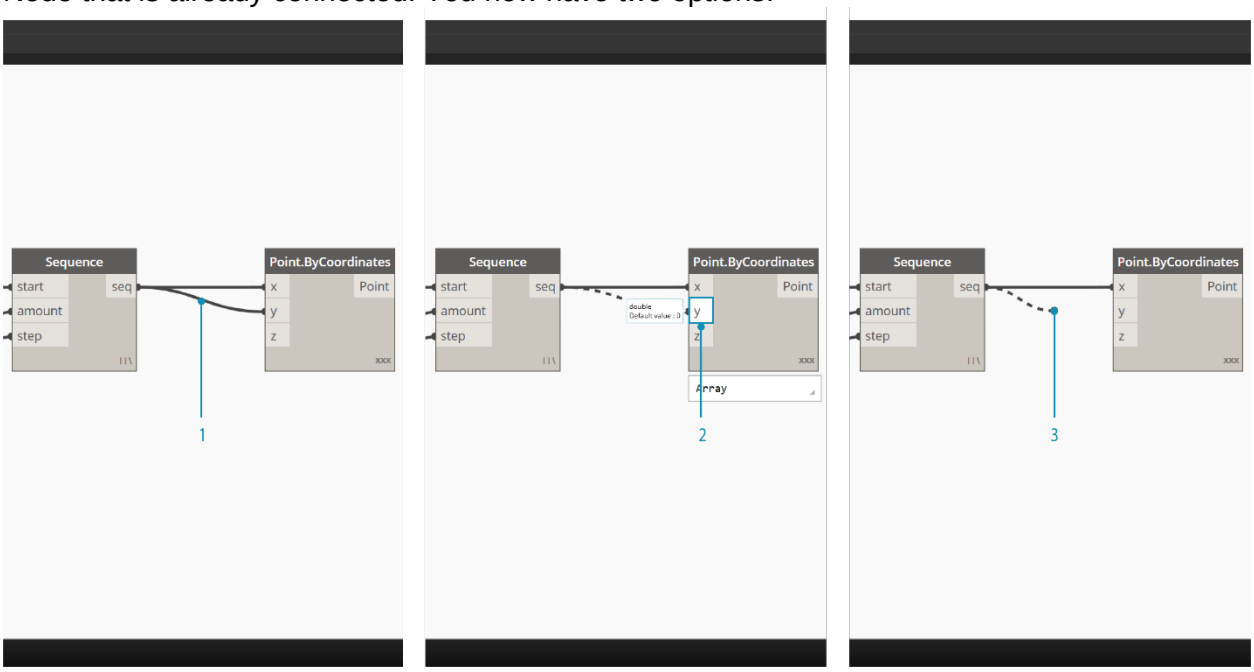
We create a Wire by left clicking our mouse on a Port and then left clicking on the port of another Node to create a connection. While we are in the process of making a connection, the Wire will appear dashed and will snap to become solid lines when successfully connected. The data will always flow through this Wire from output to input; however, we may create the wire in either direction in terms of the sequence of clicking on the connected Ports.



1. Click on the seq output Port of the Number Sequence Node
2. As you are moving your mouse towards another Port, the Wire is dashed
3. Click on the y input Port of the Point.ByCoordiantes to complete the connection

Editing Wires

Frequently we will want to adjust the Program Flow in our Visual Program by editing the connections represented by the Wires. To edit a Wire, left click on the input Port of the Node that is already connected. You now have two options:



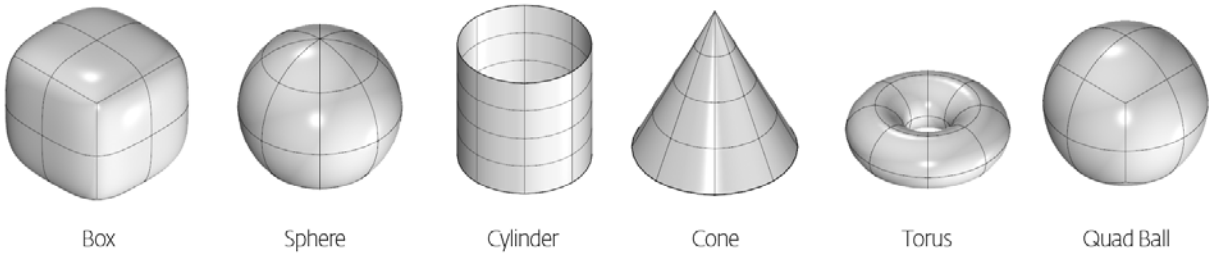
1. Existing Wire
2. To change the connection to an input Port, left click on another input Port
3. To remove the Wire, pull the Wire away and left click on the Workspace

What are T-Splines?

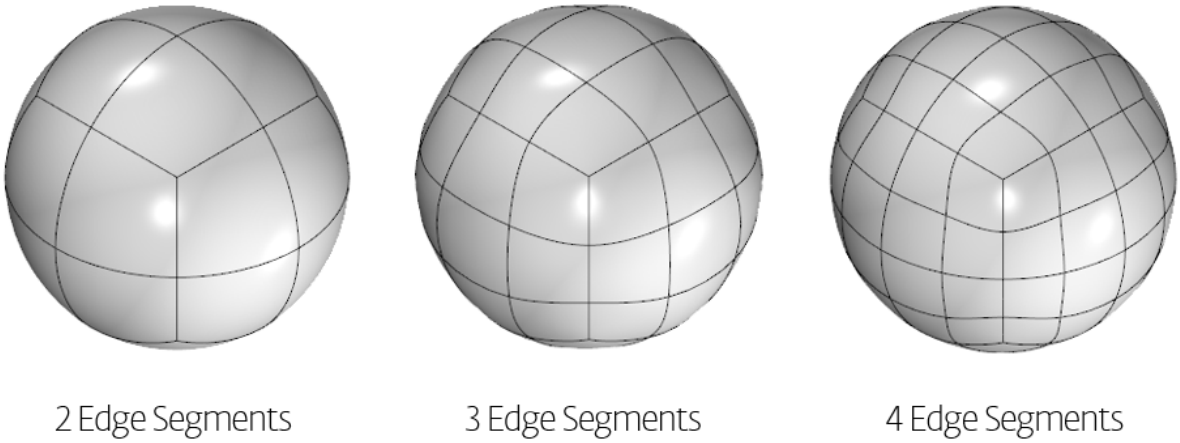
T-splines is a new mathematical formulation for surfaces that addresses the limitations of NURBS. T-Splines models are mathematically watertight, are not limited to rectangular domains, and significantly reduce the number of superfluous control points.

Working with Primitives

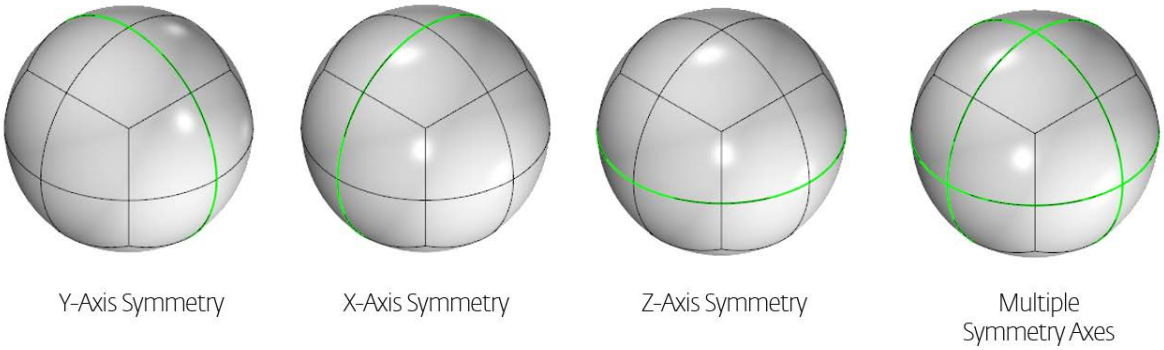
Primitives are basic geometric figures that can be edited and combined to create complex models. There are seven T-spline primitive shapes: box, plane, sphere, cylinder, cone, torus, and quadball.



A Quadball is a box-like topology with a spherical shape. When creating a Quadball, the radius and number of edge segments can be specified.

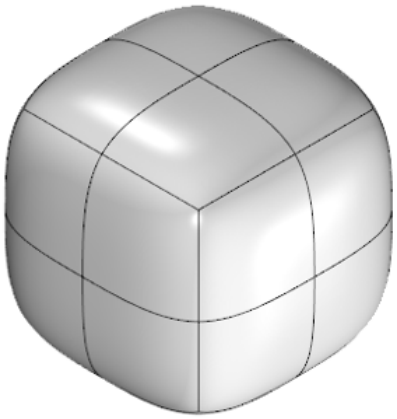


You can also select whether the Quadball should have an internal symmetry axis. Some primitives can have both axial and radial symmetry.

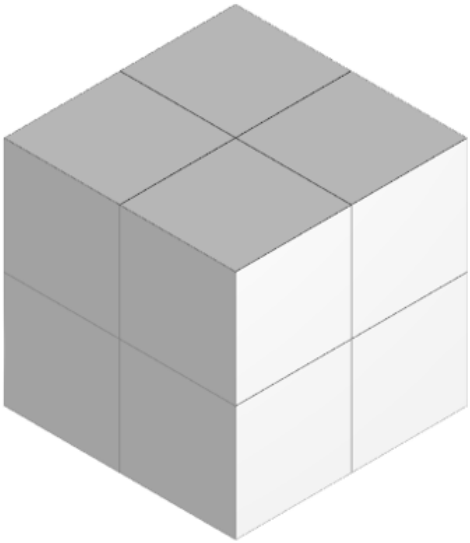


Anatomy of a T-Spline

T-spline surfaces consist of faces, edges, and vertices, and can be displayed as a boxy mesh or as a smooth surface, similar to subdivision surfaces.

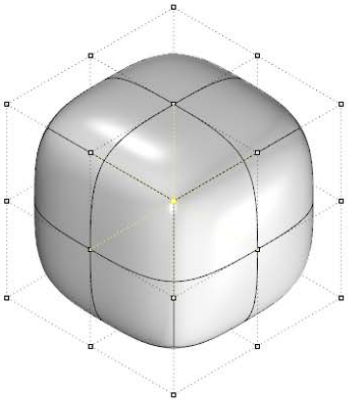


Smooth Mode

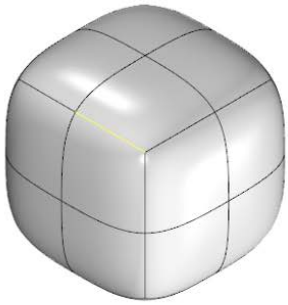


Box Mode

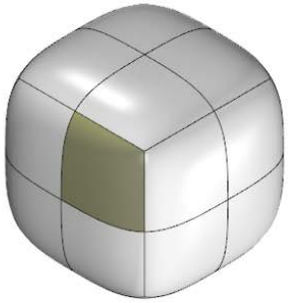
T-splines can be manipulated using vertex, edge and face grips.



Vertex Grips



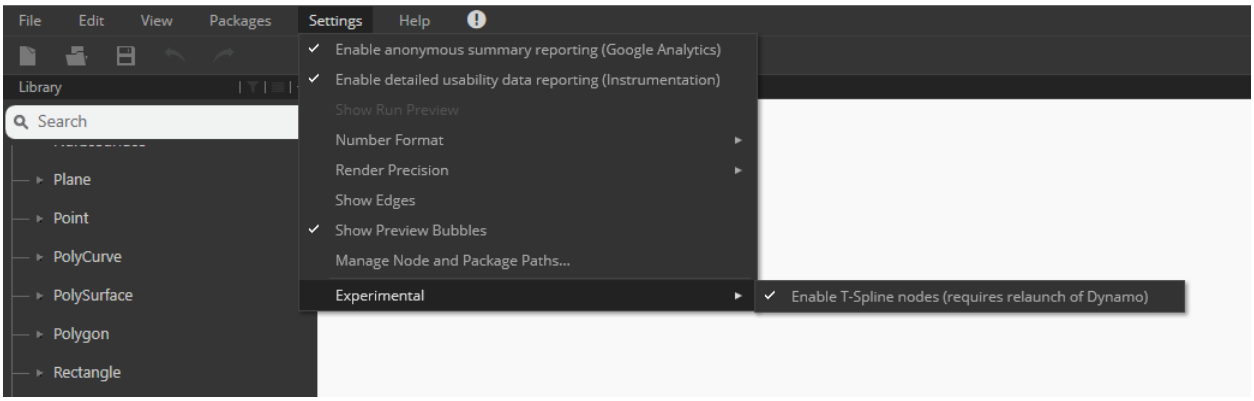
Edge Grips



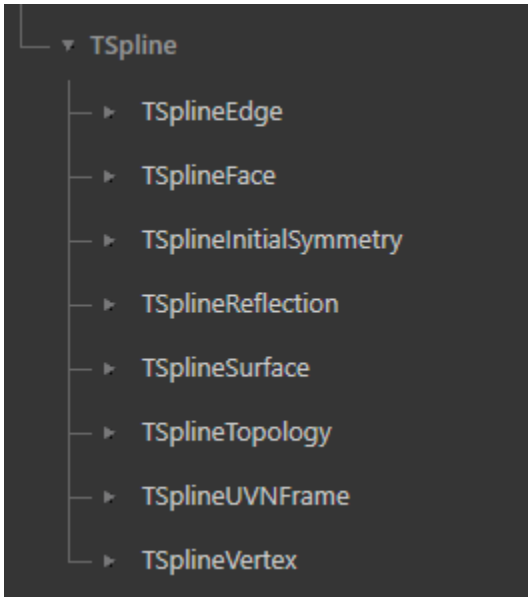
Face Grips

T-Splines in Dynamo

To enable T-Splines in Dynamo, navigate to “Settings>Experimental>Enable T-Spline nodes” and relaunch Dynamo.



You should now see the TSpline node group in the Library pane:



Extend Dynamo Functionality using the Package Manager

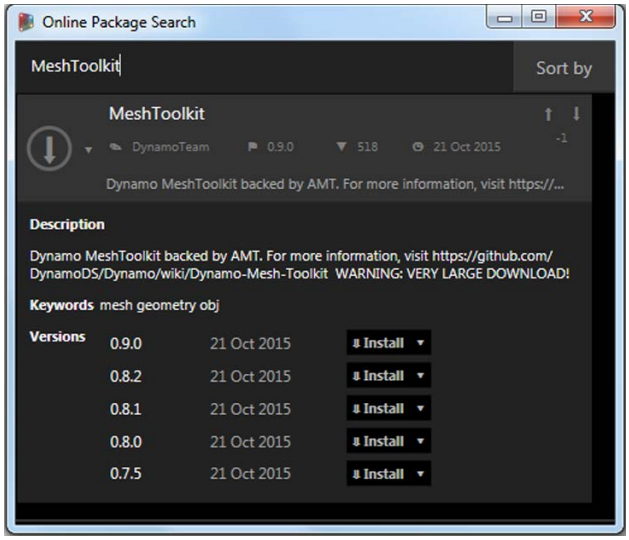
In short, a Package is a collection of Custom Nodes. The Dynamo Package Manager is a portal for the community to download any package which has been published online. These toolsets are developed by third parties to extend Dynamo's core functionality, accessible to all, and ready to download at the click of the button.

Installing a Package

To Install a Package, navigate to “Packages>Search for a Package…” Search for the package "MapToSurface" (all one word). When the package is found, click on the big download arrow to the left of the package name. This will install the package into Dynamo. After installing, the custom nodes should be available under the "DynamoPrimer" group or your Dynamo Library.

Dynamo Mesh Toolkit

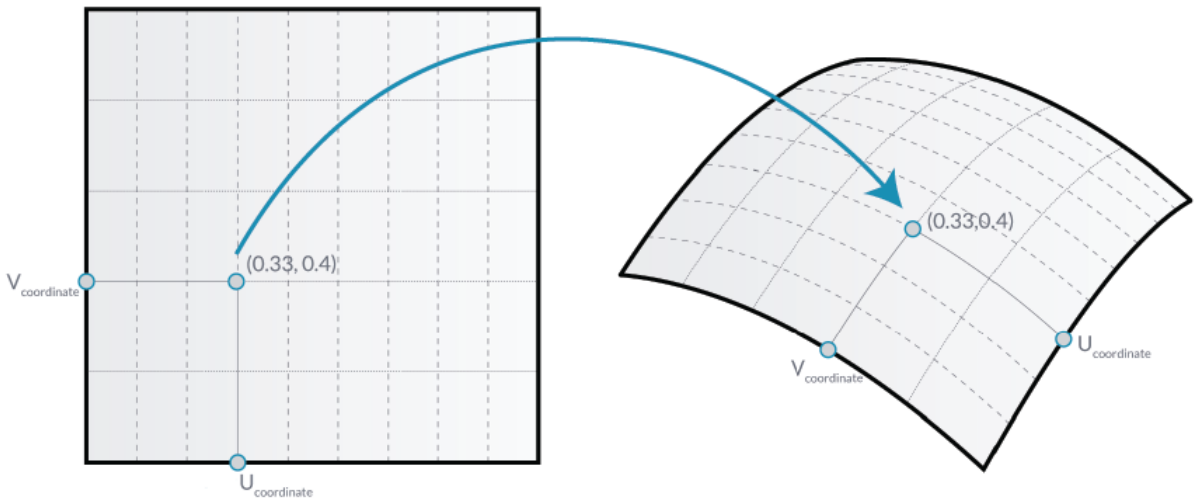
The Dynamo Mesh Toolkit provides tools to import meshes from external file formats, create a mesh from Dynamo geometry objects, and manually build meshes by their vertices and indices. The library also provides tools to modify meshes, repair meshes, or extract horizontal slices for use in fabrication.



The Dynamo Mesh Toolkit is available through the Dynamo package manager. To install, select "Search for a Package" from the Packages menu, and search for MeshToolkit. Click the down arrow in a circle icon on the left hand side of the window to install. After installation, you will have a MeshToolkit node group in the Library pane on the left side of Dynamo.

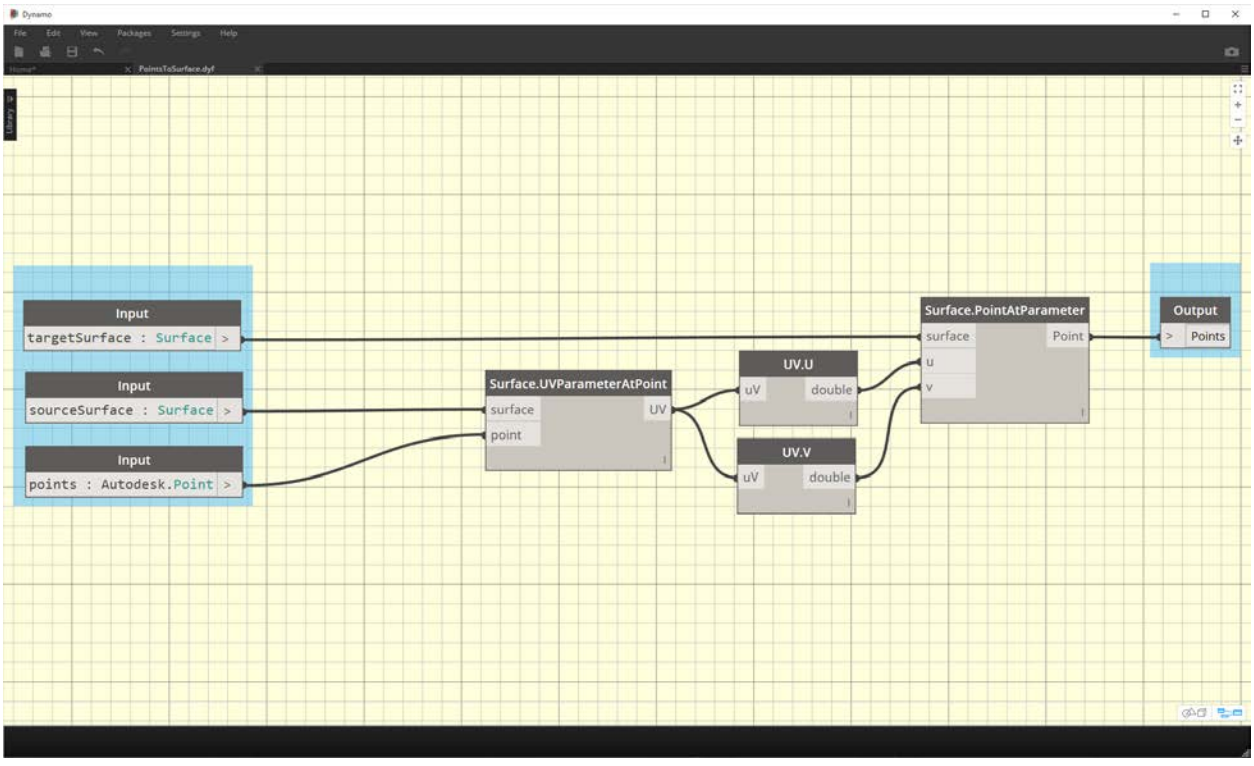
MapToSurface

Map To Surface is a Package for mapping geometry to surfaces based on UV coordinates.



In this image, we map a point from one surface to another using UV coordinates. The package is based on this concept, but with more complex geometry.

This is a simple package with five custom nodes. In the steps below, we'll briefly talk about each custom node's setup.



PointsToSurface: This is a basic custom node, and one from which all of the other mapping nodes are based. Simply put, the node maps a point from a source surface UV coordinate to the location of the target surface UV coordinate. And since points are the

most primitive geometry, from which more complex geometry is built, we can use this logic to map 2D, and even 3D geometry from one surface to another
.

PolygonsToSurface: the logic of extending mapped points from 1D geometry to 2D geometry is demonstrated simply with polygons here. Notice that we have nested the *"PointsToSurface"* node into this custom node. This way we can map the points of each polygon to the surface, and then regenerate the polygon from those mapped points. By maintaining the proper data structure (a list of lists of points), we're able to keep the polygons separate after they're reduced to a set of points.

NurbsCrvtoSurface: The same logic applies here as in the *"PolygonsToSurface"* node. But instead of mapping polygonal points, we're mapping control points of a nurbs curve.

OffsetPointsToSurface: This node gets a little more complex, but the concept is simple: like the *"PointsToSurface"* node, this node maps points from one surface to another. However, it also considers points which are not on the original source surface, gets their distance to the closest UV parameter, and maps this distance to the target surface normal at the corresponding UV coordinate. This will make more sense when looking at the example files.

SampleSrf: This is a simple node which creates a parametric surface to map from the source grid to an undulating surface in the example files.

Export a Dynamo T-Splines file for Continued Development in an External Application

T-Splines created in Dynamo can be exported to other applications in a few different ways. They can be exported as T-Splines to either TSS or TSM format, or converted to Mesh or Brep geometry for export to applications that do not support T-Splines.

Exporting a TSM File

Exporting a TSM file allows you to continue editing your T-Spline object in another application. The T-Spline retains all its characteristics including topology and symmetry axes. To export to TSM, use the **ExportToTSM** node.

Export a Dynamo File for Rapid Prototyping

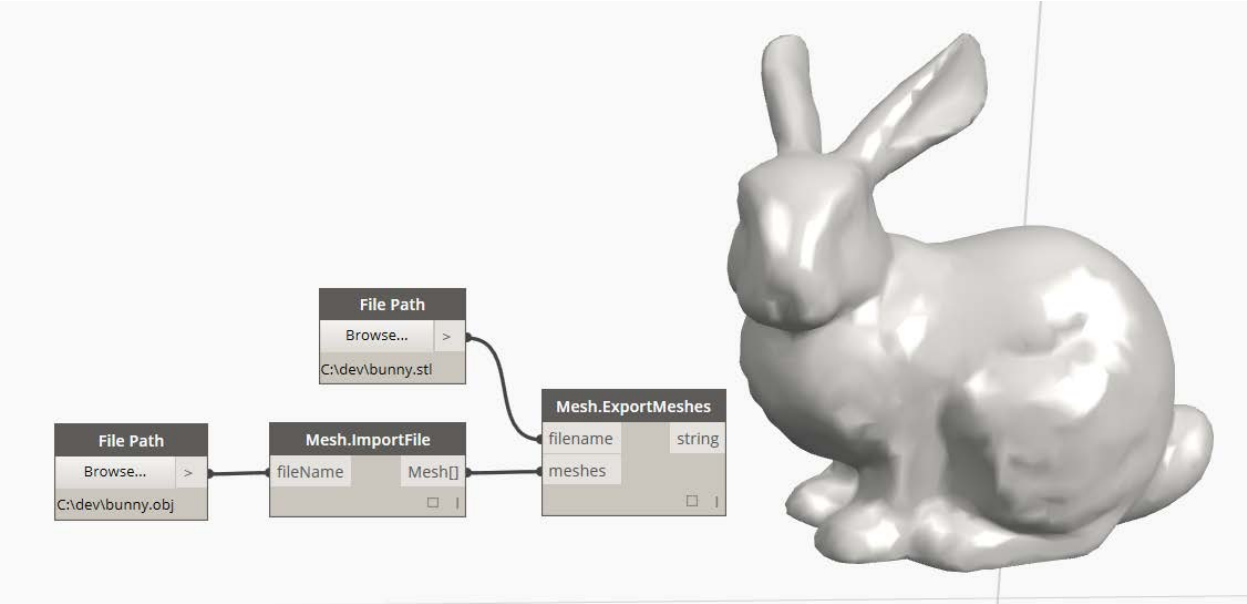
You can also convert your T-Spline to a mesh for rapid prototyping using the **ToMesh** node. Use the **isWaterTight** node first to ensure that your model is watertight. You can then perform operations such as Repair, Make Hollow, Strength Analysis, and Export from the MeshToolkit.

The MeshToolkit **ExportMeshes** node exports the mesh into a file format. The type of mesh to export is determined by the file name extension; for instance a file name ending in ".stl" will be exported in the STL format. The following formats are supported:

- .mix - Meshmixer
- .obj - Wavefront OBJ

- .stl - STL format
- .dae - COLLADA
- .ply - Polygon File Format

This image shows the conversion of an OBJ mesh into an STL:



Continued Learning

Dynamo GitHub

Dynamo is an open-source development project on Github. To contribute, check out DynamoDS https://github.com/DynamoDS/Dynamo. If you're interested in developing a Node library for Dynamo, the easiest place to start is by browsing the DynamoSamples. You can learn more about developing libraries for Dynamo on the Dynamo wiki.

Dynamo Primer

A comprehensive guide to visual programming in Autodesk Dynamo Studio. This primer is an on-going project to share the fundamentals of programming. Topics include working with computational geometry, best practices for rules-based design, cross-disciplinary programming applications, and more with the Dynamo Platform. http://dynamoprimer.com/en/

Dynamo Dictionary

a searchable database for Dynamo functionality. Here you can find explanations for nodes, sample files, and links to more information on associated workflows. This site is constantly evolving as the community continues to add more information. Like the Dynamo Primer, this dictionary is open-source - check it out on our Github page and contribute! http://dictionary.dynamobim.com/

Dynamo Introductory Training Videos

The Dynamo Introductory Videos include topics such as:

- Getting situated with Dynamo
- The anatomy of a definition
- Data management
- Computational logic
- Parametric assembly

http://dynamobim.org/learn/#videoTut

Dynamo Advanced Training Tutorials

The Dynamo Advanced Tutorials Include topics such as:

- Editing Elements in Revit and Dynamo
- Editing with Formulas
- Structural Framing
- Adaptive Components
- Excel Read and Write
- Solar Orientation

http://dynamobim.org/learn/#videoTut