SD224080

Understanding Geometry and B-Rep in Inventor and Fusion 360

Brian Ekins
Ekins Solutions, LLC
brian@EkinsSolutions.com

Learning Objectives

- Discover how surfaces and solids are represented and what B-Rep is.
- Understand how to use the Inventor and Fusion 360 APIs to query a model.
- Understand what transient or temporary B-Rep is and discover some of its capabilities.
- Understand how to create and display transient and temporary B-Rep bodies.

Description

A critical component of any CAD system is the ability to accurately describe a design. For mechanical CAD systems, the solid and surface geometry that's used to represent part geometry is extremely important. This class will investigate the internals of how Inventor and Fusion 360 represent solids and surfaces. We'll look at how solids and surfaces are constructed and modified, and how you can access this information through the Inventor and Fusion 360 APIs. We'll look at how to use the API to query and evaluate the shape of a model. We'll also cover some modeling functionality that is only available through the API. Because both Inventor and Fusion 360 both use Autodesk Shape Manager for the modeling core the APIs for both products are very similar in this area. Because of that this class can cover both products.

Speaker(s)

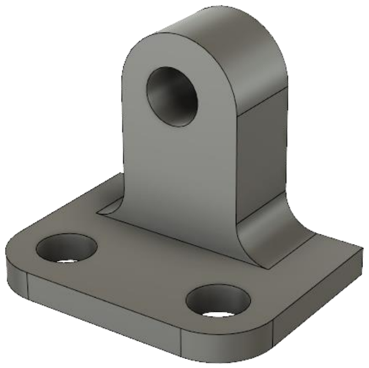
CAD has been a passion of mine ever since I saw a demonstration of an Applicon system during a high school field trip back in 1977. My experience in the industry since then has been varied. I’ve developed and taught training classes, both on modeling and programming. I’ve worked as an Application Engineer where I specialized in complex modeling and customization of the software and I demonstrated the software and performed benchmarks to show customers that the software could do what they need. When Intergraph began developing Solid Edge I moved from the role of a modeler and API user to an API designer. I then moved to Autodesk where I designed the API for Inventor and most recently, I been designed the API for Fusion 360. Now I run my own consulting/contracting business where I help Inventor and Fusion 360 users make Inventor and Fusion 360 into a customized tool specific to their needs. Contact me if you need help with your use of the software. https://EkinsSolutions.com

Introduction

Solid and surface models are at the core of Inventor and Fusion 360. A solid model is a digital representation that accurately describes a 3D object. The ability to access and fully analyze that solid model is critical for many applications. When working interactively with Inventor and Fusion 360, it is intuitive to work with a solid model because you can easily see and identify areas you want to manipulate. For example, if you want to fillet some edges of the part, it's simple to specify the edges by interactively selecting them. This paper looks at how Inventor and Fusion 360 create a model, how a model is represented internally within Inventor and Fusion 360, and how you use the API to work with that model, which is much different than using the user interface.

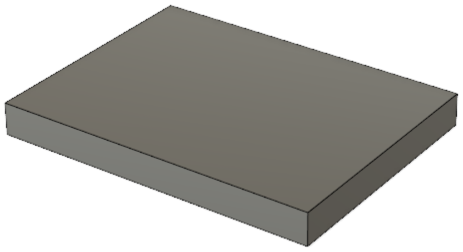
Working with models using the API is much more difficult than through the user interface but can also be more powerful by allowing you to automate routine tasks that are extremely tedious and error-prone when done manually. I like the analogy of flying an aircraft. When using Inventor and Fusion 360 interactively, it's like flying on a nice clear day with full visibility; it's easy to see where you're going and to visually follow your course. Using the API is like covering the windows of the airplane with black paper and flying entirely by instrument. It's only by knowing how to read and interpret the instruments that you'll be able to follow your course, avoid collisions, and land successfully. With Inventor and Fusion 360, it's trivial to interactively select an edge to fillet, but to specify the same edge using a program, you'll have to learn how the API provides access to the model geometry.

Building a Model Interactively

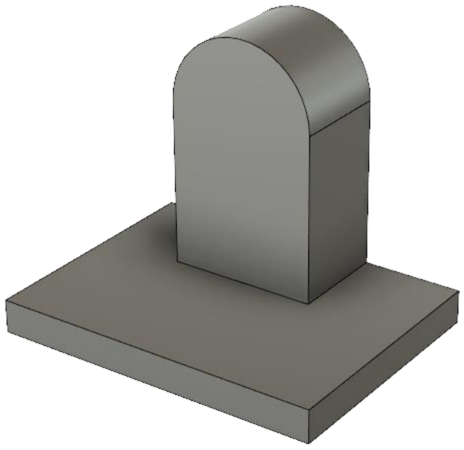


Before we look at how to access a model through the API (fly by instrument), let's review the process of creating a model interactively (fly by sight) and think about the steps you go through as we look at one way to build this model.

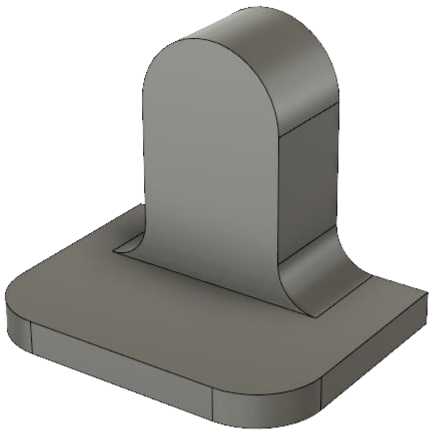
1. As with all parts, we begin by creating a sketch. In this case, the base XY construction plane has been chosen as the sketch plane, a rectangle of the correct size has been drawn and used to create an extrusion.



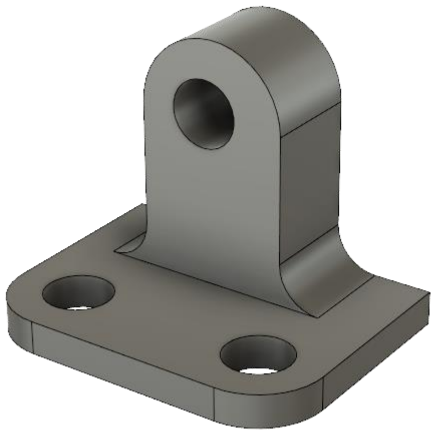
2. In the next step, another sketch is created by selecting the back face of the block. Another simple shape has been drawn and extruded to create the shape on top of the block.



3. Next, some fillet features are created to round and fillet some of the edges.



4. Finally, some hole features are placed to complete the model.



What I've just shown is a very simple model and is trivial to model for anyone with a little bit of experience with Inventor or Fusion 360. However, let's look at what must be done to create this model and the things to consider if you are writing a program to create the same design.

After creating the first extrusion, the next step is to create a sketch on the back face of the block in step 2. When working interactively, you easily select the face. In a program, it's possible to interact with the user and have them select things but you don't want to do that when you're trying to automate a process. In this case, we want to somehow find that back face without involving the user.

In step 3, you need to find the four edges that you want to fillet. And then finally in step 4, you need to find the faces that the holes will be concentric to.

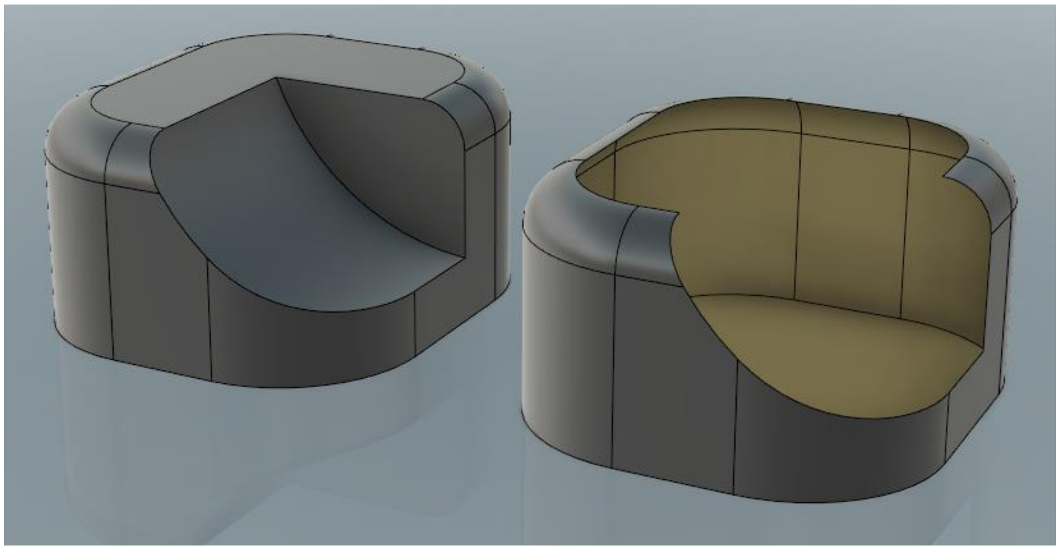
All these operations involve finding specific pieces of the existing model. This is a common use of the B-Rep portion of the API. The need to query a model isn't limited to creating models but is also used when analyzing an existing model and in assemblies when automatically creating constraints and joints. It can also be used when creating custom translators to export Inventor and Fusion 360 models into some other format.

The concepts described in the rest of this paper describe what a model actually is and how you can interrogate it to find the specific geometry you want.

Inventor and Fusion 360 Solid Models

Now we can focus on the solid model that's been created in Inventor or Fusion 360 or imported from some other system. Both Inventor and Fusion 360 use the Autodesk Shape Manager (ASM) modeling kernel to create their solid and surface models. Even though the products support some different functionality the underlying core is the same and as a result, the programming interfaces that expose the low-level geometry are very similar. The biggest difference in this area between the two products are the names of some of the API objects, which I'll point out as we go along. In fact, most of the concepts described here also apply to other non-Autodesk solid modelers.

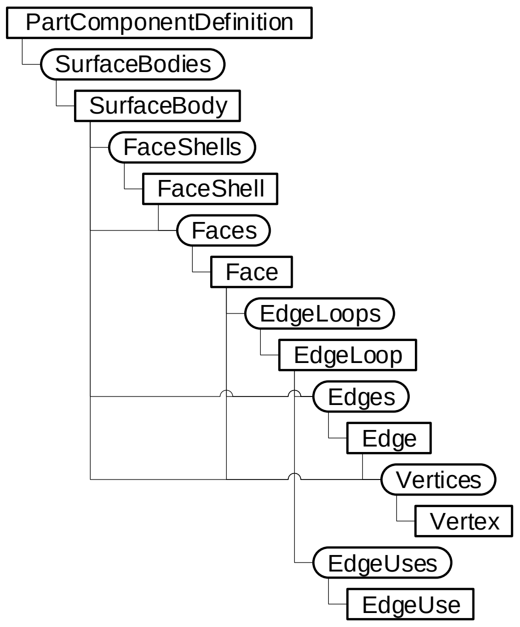
A solid model isn't really solid but is a group of surfaces that enclose a volume. This is known as a Boundary-Representation or B-Rep. A B-Rep provides a complete geometric description of a solid or surface model. A solid is a special case of B-Rep where the surfaces are tightly connected along all the edges to form a closed volume. Because of this, the modeling kernel can compute mass properties and perform other operations as if it was solid. When you create a new feature, the modeling kernel is creating the surfaces that represent that feature, intersecting them with the existing model, and trimming everything back to get a closed volume.

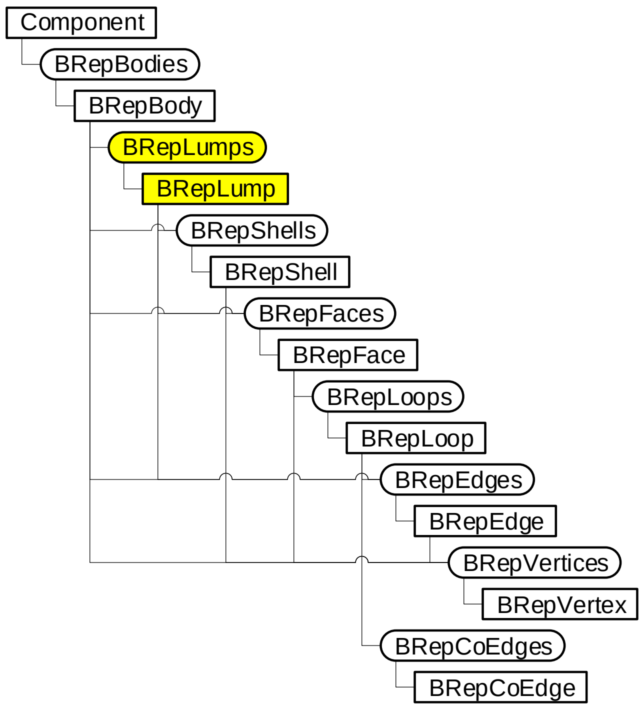


There are two concepts you need to understand to be able to work with a B-Rep model; *topology* and *geometry*, which are described in the sections below along with the API functionality that lets you access this information. An important point to understand is that the functionality described is providing read-only access to the model. It does not provide any editing capability. Editing of an Inventor or Fusion 360 model is always done through features.

Topology Defined

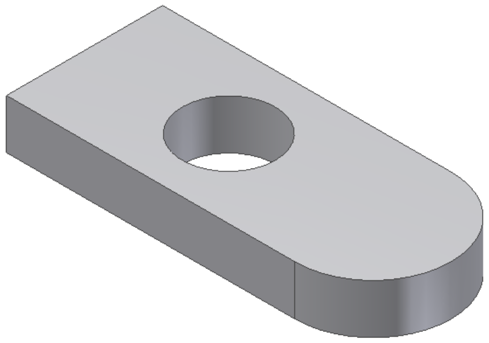
The topology defines the structure of a B-Rep model. It's a hierarchical structure of objects. The full API object hierarchy for the B-Rep topology is shown in the following illustration where Inventor is on the left and Fusion 360 is on the right. There's a one-to-one correspondence between the objects except for the BRepLump objects, which only exist in Fusion 360 and are discussed in more detail below. The discussion that follows describes each of the objects in the hierarchy. I refer to the objects using "InventorObject/Fusion360Object" notation. Fusion appends the name of all these objects with "BRep" to differentiate B-Rep objects from Mesh and T-Spline objects.

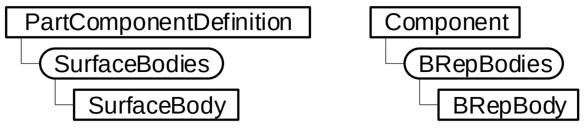




Bodies

You access the B-Rep through the PartComponentDefinition/Component object. These objects provide access to a collection object that returns all the bodies in the component. As far as the B-Rep structure, the top-level object is the SurfaceBody/BRepBody object (or commonly referred to just as “Body”). The Inventor name is a bit confusing, but despite the name, a body in both systems represents both solid and surface models. From the API hierarchy in the following illustration, you can see that SurfaceBody/BRepBody objects are accessed using the SurfaceBodies/BRepBodies collection object obtained from the PartComponentDefinition/ Component object. Most parts in Inventor have a single body, like the simple example shown below, but they can have 0, (an empty part), or more than one body. In Fusion 360 it’s not uncommon to have multiple bodies in a component.

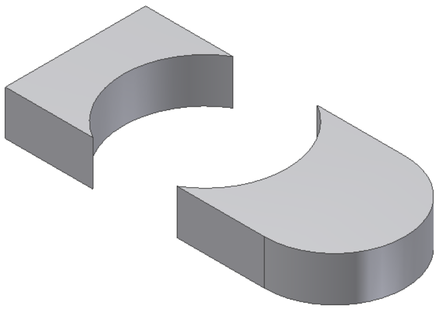


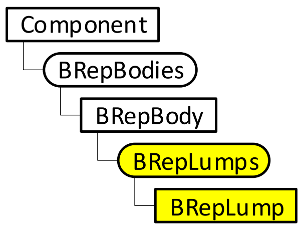


Lumps

The BrepLump object is only supported by Fusion 360. Internally within the modeling core it also exists in Inventor, but it’s not exposed through the API. It was a design decision at the time to not expose it since it’s not commonly used. In Fusion 360 it was decided that it would be best to expose the full set of internal data. The ironic thing is that Fusion 360 doesn’t allow you to create a model with multiple lumps but automatically breaks them into independent bodies so there is always one lump per body. In Inventor you can have multiple lumps, but the API doesn’t reflect that. It’s never been a problem though because even in the rare case where someone needs lump information, it can be derived using the other objects that are provided.

A lump represents a set of connected faces. In the picture below, the body has been split with a large hole. In Inventor, this is still a single body, but each of the pieces are represented internally as a lump. In Fusion 360, when the hole is cut, the result will be two bodies.

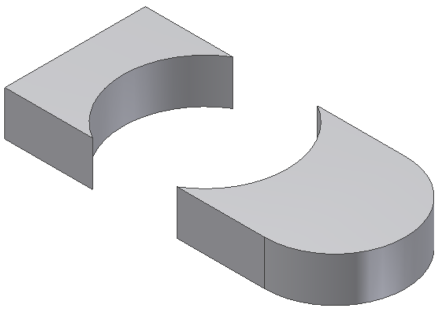


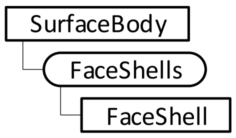


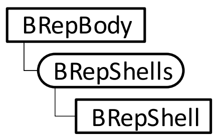
Shells

The FaceShell/BrepShell object represents a single set of connected faces. In most cases, a shell is equivalent to a lump. The exception is when you have a model that has an internal void. For example, if you create a ball and hollow it out with another ball, this model is a single body that contains a single lump and has two shells; one shell defines the outer set of faces, and the other the inner set of faces. This is extremely rare because you typically try and avoid internal voids because of manufacturability issues but it is possible to model them.

For most models, a body will have a single shell. In Inventor, the example of the body being split by a feature illustrates a body that has more than one shell. In Fusion 360, this case won't exist because Fusion 360 will automatically create a new body instead of a new shell. The only time you can have multiple shells in Fusion 360 is the case where there are internal voids.

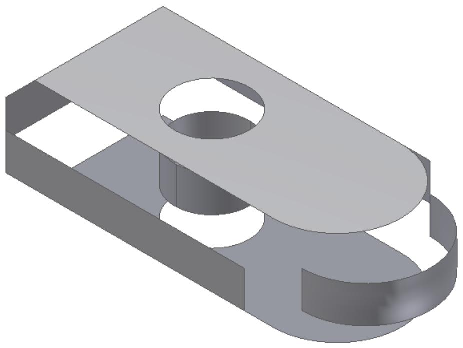


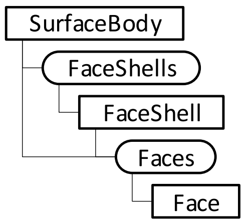


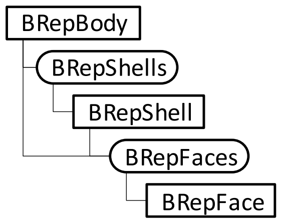


Faces

The Face/BRepFace object represents a single surface within a body. The following illustration shows an exploded view of a body, where you can clearly see each face that makes up the model. Using the B-Rep portion of the object model, you can access faces using a Faces/BRepFaces collection you obtain from either the parent body or a shell object. The Faces/BRepFaces collection of the body object returns all the faces, regardless of which shell they are part of.

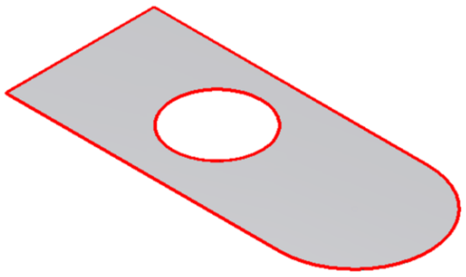


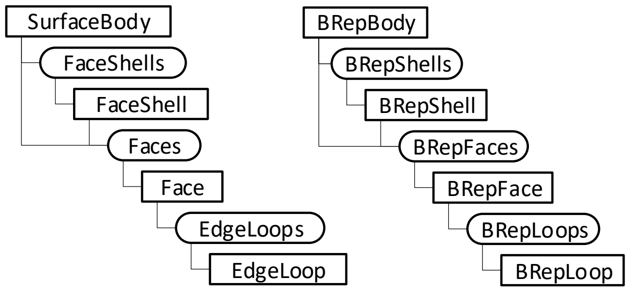




Loops

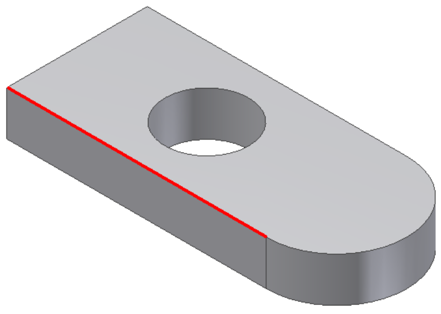
The EdgeLoop/BrepLoop object defines the boundaries for a specific face. The illustration below highlights the two loops that exist for one of the faces in the example model. All faces always have one, and only one, outer loop and can have zero or more inner loops. This example has one outer and one inner loop. The outer loop consists of four edges and the inner loop is a single circular edge.

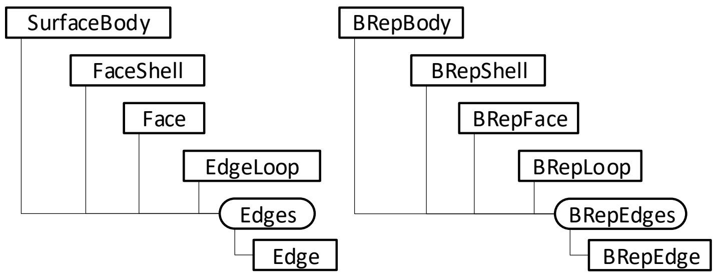




Edge

The Edge/BRepEdge object represents each curve in an edge loop. Just as important, an edge provides important topological information about the body. An edge defines the connection between two faces. Two faces share the same edge and from an edge, you can get the two faces it connects. In the picture below, a single edge is highlighted, and that one edge is shared by the two adjacent faces. For an edge along the open part of a non-solid B-Rep, the edge is connected only to a single face.

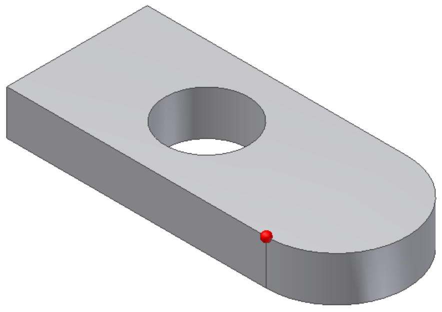


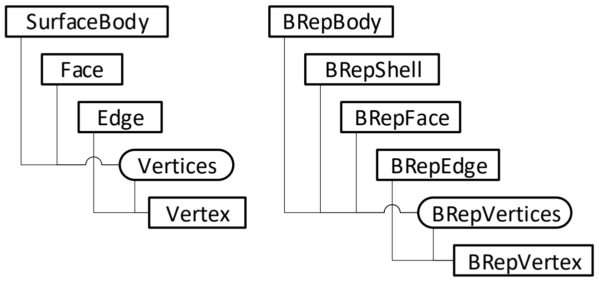


There are several ways to access edges through the API. From a body or shell object, you can get an Edges/BRepEdges collection that contains all of the edges in the body. From a Face/BRepFace object you can get the edges associated with the face; either all at once or loop by loop, using the EdgeLoop/BrepLoop object.

Vertex

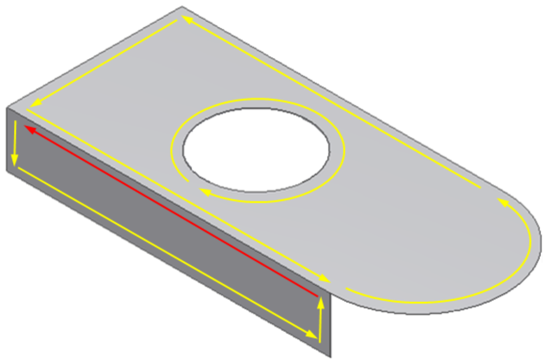
The Vertex/BRepVertex object represents a point that is at the end of every edge. The picture below highlights a vertex. In this example, the single vertex is shared by three edges. The API provides access to all the vertices in a body and face. In Fusion 360 you can also get all the vertices in a shell. Each edge returns the vertices that define its start and end. From a vertex, you can get the edges that are connected to it.

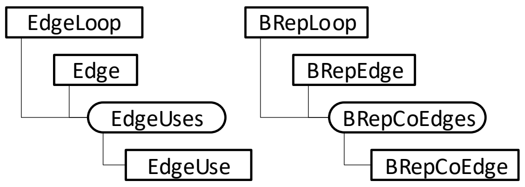




EdgeUse/BRepCoEdge

The EdgeUse/BRepCoEdge object is similar to an edge in that it defines the boundaries of a face but there are two big differences between an edge and an EdgeUse/BRepCoEdge object. The first is that EdgeUse/BRepCoEdge objects are unique for each face, whereas edges are shared between faces. One of the side effects of this difference is that EdgeUse/BRepCoEdge objects are in an ordered head-to-tail orientation around the face. EdgeUse/BRepCoEdge objects flow around the outer boundary in a counter-clockwise direction, while inner boundaries are clockwise (the material is always to the left), as shown in the following illustration. This is not possible with Edge objects because there are conflicts in direction because the edges are shared. The direction arrow in red, shown in the picture below, highlights this where the direction of the EdgeUse/BRepCoEdge for one face is in the opposite direction of the EdgeUesBRepCoEdge of the adjacent face.





The other big difference between EdgeUse/BRepCoEdge objects and edges is that the EdgeUse/BRepCoEdge object is not a 3D object. (All other B-Rep objects are 3D objects.) EdgeUse/BRepCoEdge objects are defined in the 2D parametric space of a face. The concept of parametric space is discussed in more detail below. And if this seems confusing, don't worry about it because the EdgeUse/BRepCoEdge object is rarely used.

Accessing Topology Objects

Besides traversing the object hierarchy as shown above, there are other methods available to access B-Rep objects that are much more convenient in some cases. These methods are as follows.

From Features

You can get the faces that were created by a feature using the Faces property of the PartFeature object in Inventor and the faces property of the Feature object in Fusion 360. You can get the body(s) the feature impacted by using the SurfaceBodies property of the PartFeature object in Inventor and the bodies property of the Feature object in Fusion 360.

A few of the features go further and provide categorized access to the faces they created. For example, in Inventor the ExtrudeFeature object supports the EndFaces, StartFaces, and SideFaces properties that return the end caps and the sides of an extrusion. In Fusion 360, the ExtrudeFeature object supports the endFaces, startFaces, and sideFaces properties to get the same thing. Other features provide similar functionality.

In Inventor, a flat pattern provides direct access to the top and bottom faces of the flat pattern. The Fusion 360 API doesn't currently provide any access to the flat pattern.

In Inventor, you can get the feature that caused a face to be created by using the CreatedByFeature of the Face object. This capability is missing in Fusion 360.

By Location

Another technique for finding B-Rep data is based on its position in space. There are two techniques to do this; by a point and by a ray. By a point finds the B-Rep entity that is at a specific point in space. By a ray is where you specify a point and direction in space and then the B-Rep entities that are intersected by the ray are returned, along with the point where the ray hit the entity. In Inventor, finding B-Rep data by a point is done using the FindUsingPoint method of the PartComponentDefinition. In Fusion 360 you use the findBRepUsingPoint method of the Component object.

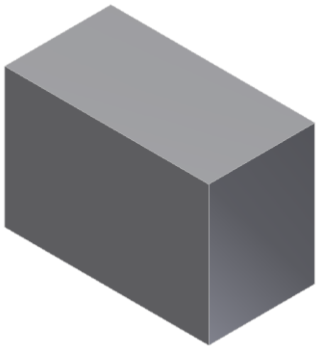
To find an object using a ray in Inventor you can use the FindUsingRay or FindUsingVector methods of the PartComponentDefinition object. FindUsingRay is limited to B-Rep entities and is more complicated to use than FindUsingVector. FindUsingVector will work for all types of objects, including B-Rep objects and it easier to use and is faster, so I recommend using it. In Fusion 360, you can use the findBRepUsingRay method of the Component object. It's limited to B-Rep but has a simpler interface like Inventor's FindUsingVector.

A useful property that a face supports is the PointOnFace property in Inventor and the pointOnFace property in Fusion 360 that returns a point that is randomly positioned on the face. It's difficult to calculate a point that is guaranteed to be on a face. For example, if you have a tube, the end of the tube is a circular face with a hole in it. The face is very thin and without knowing the shape of the face it is difficult to determine a point on the face. You'll see the PointOnFace property used in some of the samples. It's particularly useful for determining a location to calculate a normal but is useful for other things too.

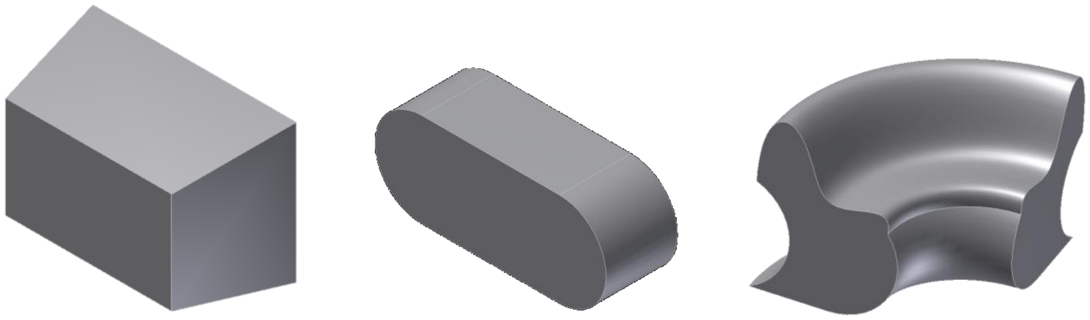
By Selection
There are many times when you cannot determine the entities needed automatically. In these cases, you need to ask the user to select them. When the user selects entities in a solid model, the objects returned are SurfaceBody/BRepBody, Face/BrepFace, Edge/BRepEdge, or Vertex/BRepVertex objects.

By Association
Another way to access a specific B-Rep entity is through its association with some other object. For example, given an assembly joint, you can obtain the two faces that that the joint is associated with. Attributes are another important tool that you can use for finding a specific entity at some later time.

Evaluating Topology Objects
Before going on it's important to understand the difference between topology and geometry. The topology defines the structure of the model and geometry defines the shape. Here's an example that helps to illustrate what topology does not provide. If you were told to describe what a model with 6 faces and 12 edges looks like, what would you describe? The obvious answer is the simple block model shown below. You would be correct in that it is a model with 6 faces and 12 edges; however, you would also be incorrect because it's just one of an infinite number of models that meet those criteria.

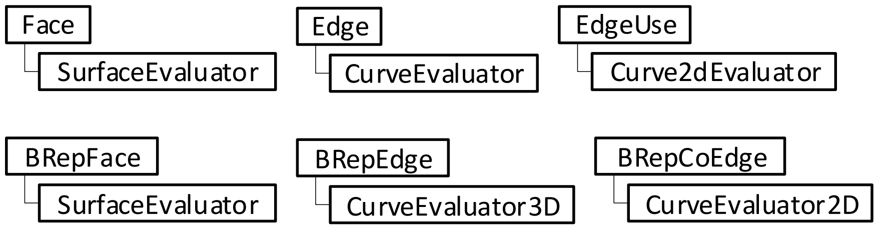


The three models shown below are examples of other models made up of 6 faces and 12 edges. All these models have the same topology but obviously have radically different geometry.



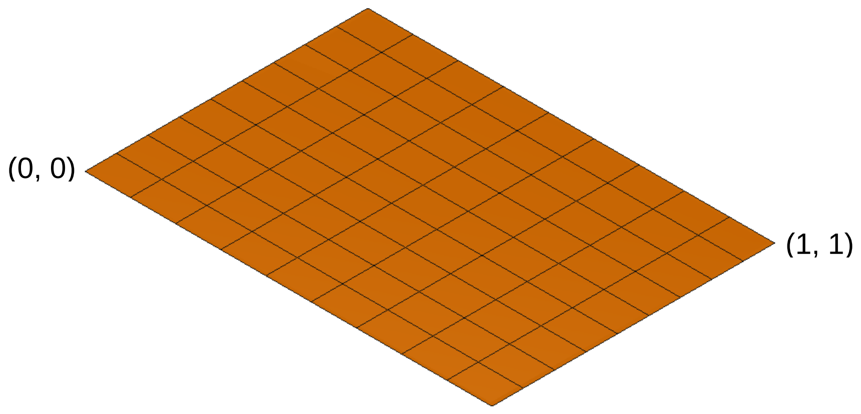
The important thing to understand is that a face represents a surface but does not define anything about the shape of that surface. The same is true of edges; they represent a curve but do not define the shape of the curve. The primary purpose of the topology objects is to define how the various geometries are connected. To fully understand the shape of the model, you need to look at the geometry associated with the topology object.

Before discussing the geometry, there are some generic queries you can perform on the B-Rep objects that provide shape related information. These queries are performed using one of the API evaluator objects, as shown here.

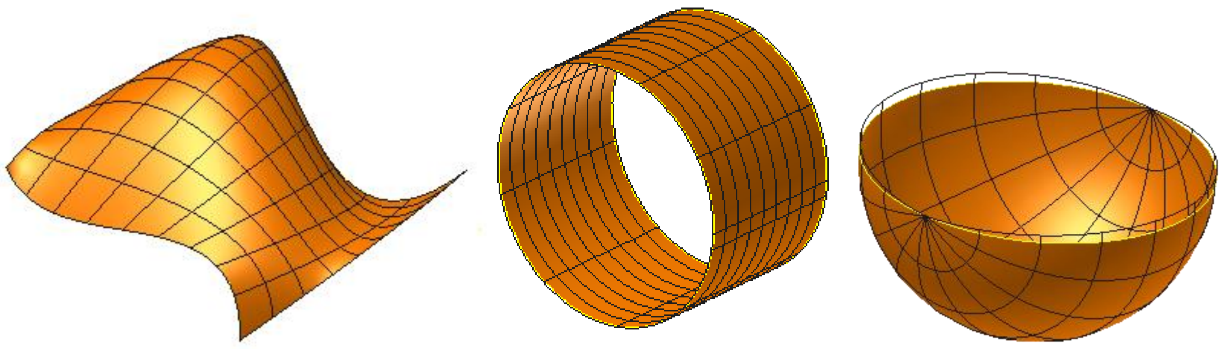


One of the first things you need to understand before you can use these evaluators is that most of the evaluations are performed relative to the *parametric space* of the surface or curve. You are used to working in model space and dealing with coordinates in its x, y, z 3D space. Parameter space is similar in that it defines a space, but for surfaces it is a 2D, like a graph paper and for curves it is a 1D space like a number line. Every surface or curve in a model has its own unique parameter space.

The picture below shows a planar face with a grid drawn on it to illustrate its parametric space. Any location on the surface can be precisely specified using two values. For parametric space, instead of x and y, the letters u and v are used to indicate the coordinates. If the minimum values are (0, 0) and the maximum are (1, 1) as shown here, then a value of (0.5, 0.5) is at the parametric center of the face.

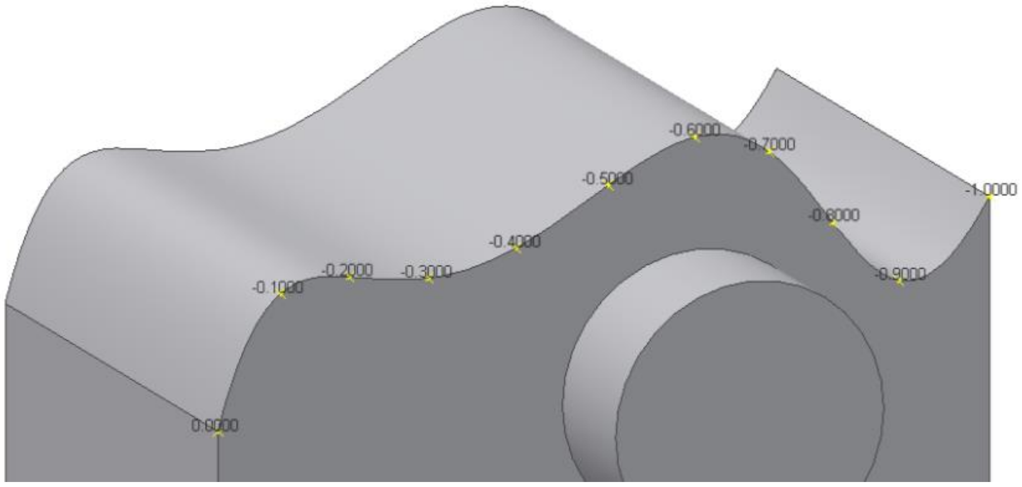


Using the previous picture it's easy to visualize how it's possible to specify any position on a plane using a u and v coordinate. Below are some examples of other surface shapes with their parameter space grid drawn on them. The surface on the left is somewhat of a variation of the plane. If you imagine the plane above is made of rubber, you could stretch it into the shape below. Any point on the surface can still be specified using a u-v coordinate. The surface in the middle can be formed by rolling up the rubber sheet into a tube. Again, two values can define any point on the surface. Finally, with the surface on the right, two of the edges of the original rectangle have been shrunk down to zero length, but still, a u-v value defines any point on the surface.



Why should you care about the parameter space of a surface? Usually, you don't; you typically only need to know about things relative to 3D model space. However, the functions that perform the surface evaluations work in parametric space, so you'll need to become familiar with it in order to use the evaluators effectively.

The parameter space of an Edge is a one-dimensional space. Any point on the edge can be identified with a single parameter value. The start and end parameter values are the extents of the curve, and any value in between represents a unique point along the edge, as shown in the illustration below.



To give you an idea what the evaluators support, below is a list of some of the more commonly used functions, but there are others too.

SurfaceEvaluator

Area – Returns the area in square cm^2

GetNormal – Calculates the normal vector of the face at a specified parameter point. The normal always points out of the solid.

GetParamAtPoint – Given a 3D model point this returns the equivalent 2D parametric point.

GetPointAtParam – Given a 2D parametric point, this returns the equivalent 3D point.

IsParamOnFace – Given a 2D parametric point, this indicates if the point lies on the face or not. This is particularly useful to find out if a given point lies within a hole.

ParamRangeRect – Returns the maximum and minimum parameter space coordinates of the face.

CurveEvaluator

GetEndPoints – Gets the start and end points of the edge.

GetLengthAtParam – Returns the actual length of the edge between two input parameters.

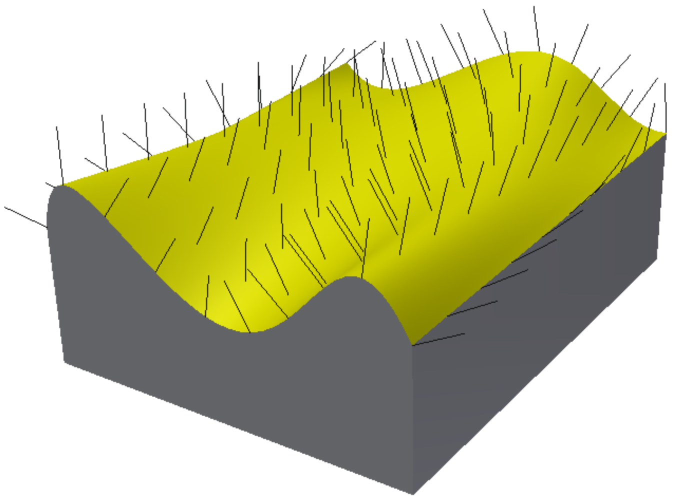
GetParamAtLength – Returns the parameter value as measured along the curve from a specified parameter point.

GetParamAtPoint – Given a 3D model point this returns the equivalent parametric value along the edge.

GetPointAtParam – Given a parametric value, this returns the equivalent 3D point.

GetParamExtents – Returns the minimum and maximum parameter values of the edge.

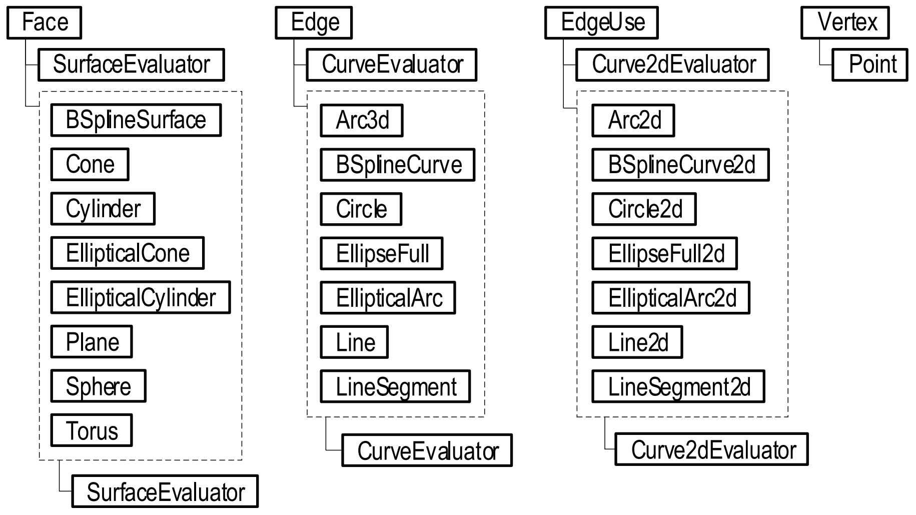
Two of the most commonly used functions of the SurfaceEvaluator is the ability to calculate the area and get surface normals. Using the Area property is easy, since it doesn't require any arguments, and returns the area of the face in cm^2. A normal is a direction that is perpendicular to a face at a specific point. A planar face has the same normal at any location on the face, whereas a spherical face has a different normal for every position on the face. For a solid, the normal always points out of the solid. The picture to the right shows a series of normals displayed on a spline face. You can see how they are perpendicular to the face and how they all point out of the solid. Getting normals is often used as a way to determine the orientation of geometry and is one way to differentiate one face from another.



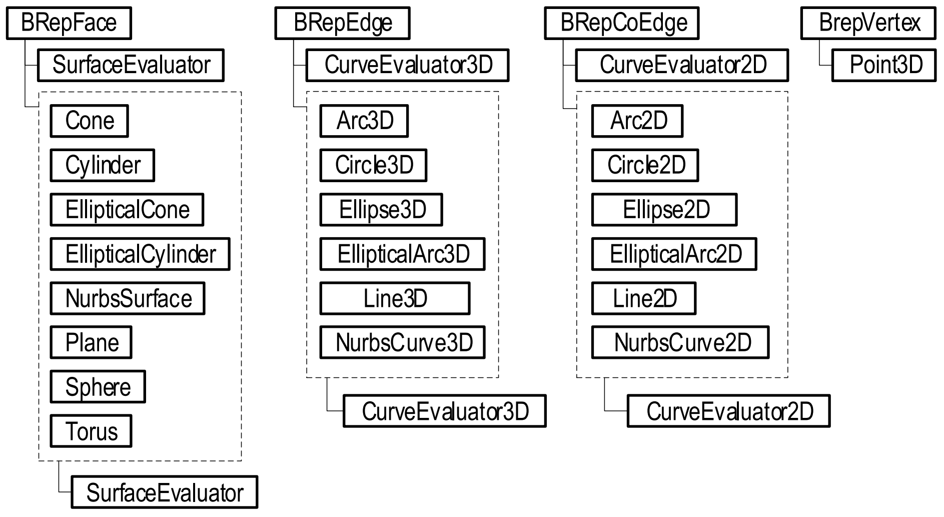
Geometry

We've seen from the previous discussion that the topology itself does not fully describe the shape of the model but defines the structure or how the model is connected. To fully understand the shape of the model, you use the geometry of each face and edge by using the Geometry/geometry property of the face, edge, or EdgeUse/BRepCoEdge object. The geometry property returns one of several objects, depending on what the actual shape of the Face/BrepFace, Edge/BRepEdge, or EdgeUse/BRepCoEdge object is. The following illustration show the types of geometry objects that can be returned for each of the B-Rep types.

Inventor



Fusion 360



In Inventor, in addition to the Geometry property, the Face object also supports the SurfaceType property, and the Edge and EdgeUse objects support the GeometryType property. These properties return an enum value that specifies the type of geometry represented by the B-Rep entity. It’s an easy way to determine what type of object the geometry property will return. In Fusion 360 you need to get the geometry and then check the type of object returned.

You might notice in the previous diagram that you can get a SurfaceEvaluator from both a face and from the various geometry objects. If you have access to a face or edge, it’s better to use its evaluator because it considers the rest of the body. For example, if you get a normal from a geometry object, the normal is not guaranteed to point out of the solid, since the geometry doesn’t know anything about the solid.

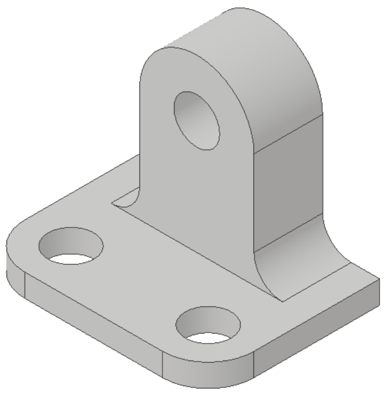
Each geometry object provides properties that describe the shape of the geometry. For example, a Cylinder object supports properties to get its base point, axis vector, and radius, which fully defines a cylinder. A Plane object supports properties to get its position and normal vector. One thing you might notice from these properties is that the geometry they are defining is without bounds. The cylinder has infinite length in both directions, and the plane is infinite in all directions. It’s the B-Rep, specifically the edge loops, that define the boundaries of the geometry.

Something else you need to be aware of when working with the various geometry objects is that they are a “tear off” from the B-Rep entity. This means that once you get the geometry object, there is no relationship back to the B-Rep entity you got it from. For example, if you get a Cylinder object from a face, the cylinder accurately describes the shape of the face, but if you modify the model in a way that causes the face to change, the Cylinder object does not reflect that change. The cylinder you have is a snapshot of the geometry at the point you got it. The cylinder and the face are completely independent. To get the updated version of the geometry you need to get a new Cylinder object from the face. Similarly, if you change the radius of the cylinder it does not affect the face in any way.

Putting it Into Practice

So far there’s been a lot of theoretical discussion without much practical application. To best illustrate these concepts, I have some sample programs to demonstrate their use. Here is a detailed description of some of these programs and how they use the B-Rep and geometry data to accomplish the desired task.

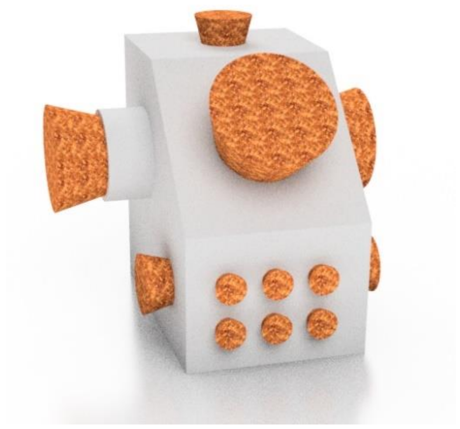
Model a Part



This program creates a part from scratch and demonstrates a few of the techniques for finding specific B-Rep entities that have been discussed. It finds a face based on its position in model space and some edges based on their position and orientation. It finds cylinders based on face geometry and finds edges by looking for common edges between a feature and a known face. When looking for specific geometry, there isn’t a right or wrong way to do it, as long as you find something that is reliable. Some solutions may be more efficient that others but, in most cases, it probably isn’t going to make much difference in the overall speed of your program.

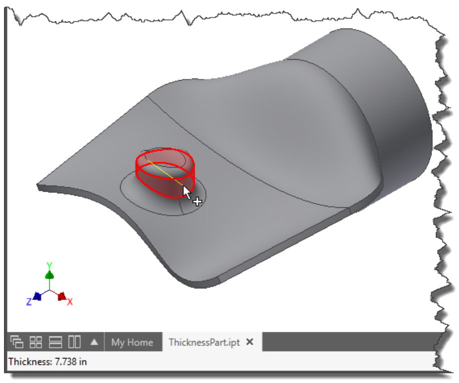
Corks

This sample takes any selected part and inserts a cork into every hole. It does this by using much of what has been discussed. It traverses through all the faces of the part, looking for planar faces. For each planar face it looks at all the loops, looking for interior loops that consist of a single circle. It then finds the connecting face, which will be a cylinder and uses the normal of that face to determine if it represents a hole or a boss. It ends up finding all the edges of holes in the part. It then copies and edits an existing cork part to make a cork of the correct size and inserts it into the assembly. Finally, it creates a constraint between each cork and the edge of the hole it fits in.



Measure Thickness

This is an Inventor only sample that displays the thickness of the part as measured at the current position of the mouse. There's some functionality missing in the Fusion 360 API that makes it difficult to get the current mouse position on the face.

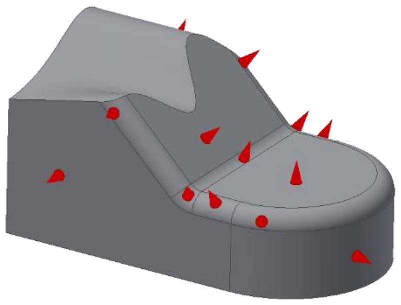


Thickness: 7.738 in

This sample uses Inventor's InteractionEvents functionality to start a command and get the current position of the mouse as it moves over the faces of the model. With that coordinate point, it then gets the normal of the face and fires a ray through the model to find the next face along the normal and the point on that face. The distance between the two points is the thickness of the part. It uses interaction graphics, which is a type of client graphics, to display a line showing how the thickness is measured and displays the thickness in the status bar.

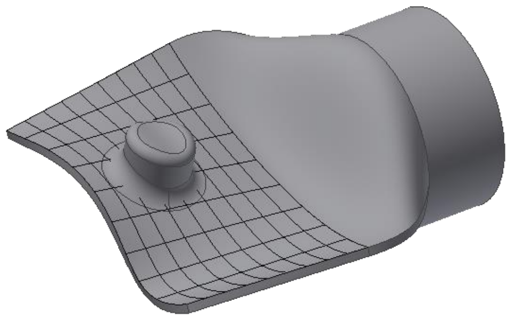
Show Body Normals

This sample displays a small cone on each face of a selected body, showing the normal direction of that face. To do this it traverses over the body to get each face and then uses the PointOnFace functionality to get a point somewhere on the face and uses that to calculate the normal. It also uses the RangeBox property of the body to get the overall size of the part and uses that to determine the size of the arrows, which is draws by creating transient/temporary B-Rep cones and displays them using client/custom graphics.



UV Parametric Grid

This sample draws a grid on a selected face to illustrate the parametris space of the face. This sample demonstrates using several functions on the SurfaceEvaluator to get information about the parametric space of the face and then to convert that information into model space coordinates. Finally, it displays the result using Client/Custom graphics.



Alternate Representations

You can get model information in other forms besides the B-Rep model. One option is to get a body that consists of a modified version of the original body. The most common use of this method is to get a body that is completely made of NURBS (Non-Uniform Rational B-Spline) surfaces. Some applications work exclusively with NURBS since almost all geometry can be precisely represented as a NURBS surface. In Inventor, you can use the AlternateBody property of the SurfaceBody object. This property returns a modified SurfaceBody that meets the criteria specified. The following example shows the most common use of this property.

```
Dim body As SurfaceBody = partDef.surfaceBodies.item(1)

Dim nurbsBody As SurfaceBody
nurbsBody = body.AlternateBody(SurfaceGeometryFormEnum.SurfaceGeometryForm_NURBS Or _
                         SurfaceGeometryFormEnum.SurfaceGeometryForm_ProceduralToNURBS)
```

In Fusion 360, you can use the convert method of the BRepBody object. The following is an example.

```
body = rootComp.brepBodies.item(0)
nurbsBody = body.convert(BRepConvertOptions.ProceduralToNURBSConversion)
```

One important thing to understand in this case is the body returned in both cases is a transient or temporary body. It's not created within the Inventor or Fusion 360 document but only exists as an API object and is gone as soon as the variable referencing it goes out of scope. Both Inventor and Fusion 360 support some additional capabilities with transient/temporary B-Rep, which is described below.

Transient/Temporary B-Rep

Both Inventor and Fusion 360 support the ability to create and modify B-Rep data that is independent of any document. Inventor refers to this as "transient" and Fusion 360 uses the term "temporary" but they mean the same thing in this case. Being temporary, these objects are not associated with a document, are not visible, and are not saved. They are B-Rep entities that you can create and modify "off to the side" without affecting any Inventor data.

In Inventor, this functionality is accessed using the TransientBRep object, which is obtained using the TransientBRep property of the Application object. The fact that you get the object from the Application object implies that it's not associated with any document, which is true.

In Fusion 360, this functionality is accessed using the TemporaryBRepManager object which is a static class that you can get using it's `get` property.

There are several ways to create a transient body. In Inventor, you use the Copy method of the TransientBRep object to create a copy of a parametric body. In Fusion 360 you can use the copy method of the TemporaryBRepManager. You can use one of several methods on the TransientBRep or TemporaryBRepManager object to create primitive shapes like blocks, cones, and spheres. You can import geometry from external sources using the ReadFromFile method in Inventor and the createFromFile method in Fusion 360. You can modify a transient body by performing Boolean operations between bodies, deleting faces, and transforming a body.

Using transient/temporary B-Rep can have some big advantages over standard Inventor and Fusion 360 modeling. Because it is transient, the operations are much faster because you don't have any of the overhead that you have when creating a part using features. When building up the solid body using features, both Inventor and Fusion 360 track everything for associativity, operations are transacted so they can be undone, and graphics are created so you can see the result. None of this happens with transient B-Rep objects.

Here are some examples of the uses of transient B-Rep:

- To perform some calculations that require simple modifications of the solid. For example, you can copy the existing part body to create a transient body and then cut sections of the solid and get the volume of each section.
- You can create a solid using temporary B-Rep and then import it into Inventor to use for subsequent processing. Temporary B-Rep bodies can be imported into Inventor as a NonParameticBaseFeature. This is the same type of feature that's created when you import a foreign file like a SAT or STEP file.
- You can display a temporary B-Rep body as client/custom graphics. For commands that have little 3-D widgets made of client/custom graphics, it's often easiest to construct the geometry that will be used for the widget using temporary B-Rep and then use client/custom graphics to display the body.

B-Rep as a Mesh

Another representation of a B-Rep body is a mesh. Inventor and Fusion 360 maintain a triangular mesh of the body that they use for the graphics display of the model. You can get this existing mesh or create one of your own at any level of accuracy. Below is a picture of a mesh approximation of a B-Rep model.

In Inventor, this capability is exposed through the CalculateFacets and GetExistingFacets methods of the SurfaceBody and Face objects. In Fusion 360 you use the MeshManager object that you get from the BRepBody, BRepFace, BRepLump, and BRepShell objects.

There is also similar functionality for edges, where you can get an edge as a series of points. In Inventor this is done using the CalculateStrokes and GetExistingStrokes methods of the SurfaceBody, Face, and Edge objects. In Fusion 360 you use the calculateStrokes method of the CurveEvaluator3D object.