**Implementation Guide**

**Recent Revision: 2014-5-20 Inhwan Lee**

1. Outline
2. TransCAD example
3. Inventor example(API)
4. SolidWorks example(Macro)
5. Translation Method
6. UG Example (Macro)
7. **Outline**

This document is written to help those who joined iCAD lab for the implementation of Macro-parametric project.

TransCAD is the commonly used program in this project. You can find the detail explanations of TransCAD from TransCAD Automation API Guideline, Automation API reference document, and Handbook. In this document, we will help your understanding about implementation by showing you the example code of simple model such as L-block in both CAD system and TransCAD.

We should implement a program that translates part information from various CAD systems to TransCAD and also vice versa (from TransCAD to various CAD systems). CAD systems can be divided into two groups, one that uses API (Inventor, Pro-E, UG) and the other uses Macro (Solidworks, CATIA, UG). In chapter 2, this document will introduce the code to draw shape in TransCAD. In chapter 3, the code to draw shape using API will be introduced. In chapter 4, the code to draw shape using Macro will be introduced. In chapter 5, this document will explain the method of translation. Finally, the setup and the usage guide of UG translator is explained in chapter 6.

We wish that readers shall obtain much help for the implementation of this project from this document.

1. **TransCAD Source Code Example and Explanation**

(For the entire source code, please refer to the last part of this material.)

|  |
| --- |
| using namespace TransCAD;  IDocumentsPtr spDocuments = m\_spApplication->Documents;  IPartDocumentPtr spDocument = spDocuments->AddPartDocument();  IPartPtr spPart = spDocument->Part; |

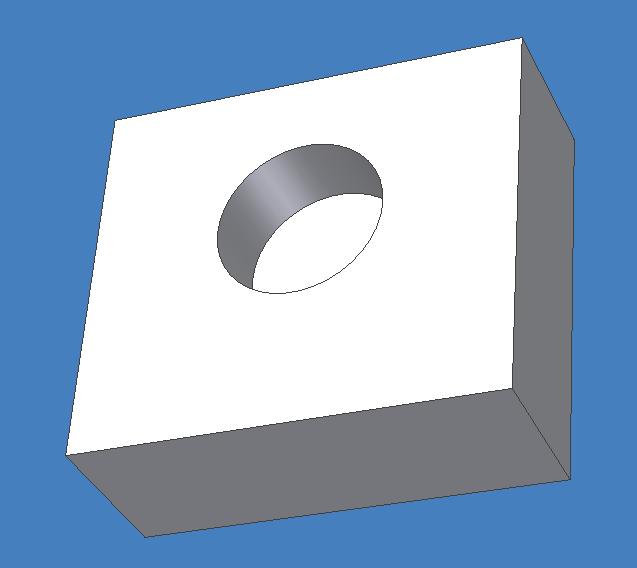
The source code shown above is to create L-Block model in TransCAD system. In the following section, the source code will be explained in detail.

The first part is a kind of initialization for generating a new part model in TransCAD, for example, if we want to write document in Word, you have to start the Word program first, then open new document. (This process is similar not only in TransCAD, but also in other CAD systems.)

If the preparation work for using TransCAD is done properly, then m\_sApplication should be declared. As shown in the source code, spDocuments, which has data type of DocumentsPtr, is declared with the m\_sApplication->Documents. In other words, the Documents of TransCAD is declared as spDocuments (here, m\_spApplicaiton refers to TransCAD.), thus, “one new document” of TransCAD is created.

If you have an overview about understanding the CAD system, it is easy to understand from the next line of source code. Even if each CAD system has its own characteristics, all CAD systems are similar. Basically speaking, if we want to create one document in a certain CAD system (here, we use document for convenience), we have to specify the document type first, for example, whether it is a part or an assembly. (Assembly consists of several Parts). Then, according to document type, the next working item is determined. Here, we assume that we want to draw one Part model, hence we will use spDocument->AddPartDocument() to declare the spDocument. If we want to draw one Assembly model, then we should use AddAssemblyDocument() instead.

Now, think about the word file example, we already knew that the source code mentioned above about preparation of job to create a new document is enough, but, one more important thing is to declare the Part first, before using API to create the Part model. Therefore, we use spPart to declare the spDocument->Part, its type is IPartPtr.



It is simple. Draw a rectangle and extrude it. Then select one face of the box and draw a circle on that face. Finally, make a hole using that circle.

In short, this process is like this:

|  |
| --- |
| Reference selection 🡪 Rectangle drawing🡪 Creation of “extrude feature”🡪 Face selection 🡪 Circle drawing 🡪 Creation of “cut extrude feature” |

At the first line of code, “spPart->Features” should be assigned to the variable ”spFeatures ” type of which is “IFeaturesPtr”. This “spFeatures” is the pointer that includes informations about “Features” in “spPart “. Like “spPart” is assigned first nevertheless nothing is drawn, this process is also an advance preparation to create “Feature”.

In the next line, “spPart->SelectObjectByName(“XYPlane”)” is assigned to the “spXYPlane”? And which type is “spXYPlane”? IReferencePtr. In other words, Selecting Reference is the first one of three steps. Let`s study step by step.

TransCAD automatically makes something when spPart which has no object is created. One of them is basic “reference plane”, a drawing paper for the first sketch. Three basic “reference plane” of TransCAD are “XYPlane”, “YZPlane”, “ZXPlane”. (You can check all of these at the left side of window.) “XYPlane”, one of these reference planes, is the selected one using “SelectObjectByName” function.

At first, let’s find out how the AddNewSketchFeature function works. According to API reference document, AddNewSketchFeature in the IFeature class makes new SketchFeature object and add it. Name and sketch plane must be written as input variables(We can return SketchFeature object as a output variable, but we don’t need it at this example).

|  |  |
| --- | --- |
| * AddNewSketchFeature   Make new SketchFeature object and add it.   |  | | --- | | HRESULT AddNewSketchFeature([in] BSTR name, [in] IReference\* pSketchPlane, [out,retval] IStdSketchFeature\*\* ppVal); |  * Parameter   pSketchPlane : the plane that sketches are drew.  ppVal : new SketchFeature object. |

In actual code, we made new sketch feature with name “Sketch1”, and the plane sketch will be drawn is spXYPlane, this input variable is the XYPlane that we declared as reference object.

Next line is ISketchEditorPtr spEditor1 = spSketch1->OpenEditor(); .To draw a sketch, TransCAD uses SketchEditor object. And to use this SketchEditor, we have to operate OpenEditor() function first.

This is the basis of sketch drawing. In summary,

1. Choose the plane that the sketch will be drawn by Reference object.
2. Declare new sketch object by input name and reference into AddNewSketchFeature function.
3. Operate OpenEditor() function of sketch object to prepare drawing.

Next 6 lines of code will make the sketch visible in TransCAD. You can sketch Arc, Line, Circle, etc. Example code above are used to Create2DLine2Points function. That is, design L-shaped sketch by using the function that drawing the line connecting two points. After that using Close() function closes sketch editor. Then you can see that L-shaped sketch appears in TransCAD.

|  |
| --- |
| bstr\_t name1 = spSketch1->Name;  IReferencePtr spBaseSketch = spPart->SelectObjectByName(name1);  // Create a protrusion extrude feature with the sketch  spFeatures->AddNewSolidProtrusionExtrudeFeature("Extrude1", spBaseSketch, 100, Blind, 0, Blind, false); |

Code is very short and simple, in fact, this part is very tricky part in the whole translation process ^^. But it's only an example, so don't worry much about that.

First string is *bstr\_t name1 = spSketch1->Name;* in this string you can see *spSketch1->Name* what's this? Do You know? Right before this process, we used *AddNewSketchFeature function* to create a *spSketch1* in that process, we make the input name of the sketch, so we can get the name of sketch using *spSketch1->Name* syntax. And here we stored the name into variable *name1.*

After that we declared a variable *spBaseSketch* whose type is *IReferencePtr* and we coded *spPart->SelectObjectByName(name1)* for its input*;* As I mentioned before, *spPart* is a class that will manage whole part model in a document that is in the TransCAD so that we can get the information of sketch and reference from this class.

And then, can you guess the functionality of *SlectObjectByName* function? We can easily guess the functionality from its name ^^. It returns the object that has same name with input string. You remember that we set the name as *Sketch1* so now we know variable *spBaseSketch* is a reference type variable that represents *Sketch1*

Next string explains how reference can be used in a code, *spFeature->AddNewSolidProtrusionExtrudeFeature* is used here.

*spFeature* is used several times so far. Yes, it's a class that manages Features of documents in TransCAD. This class has *AddNewSolidProtrusionExtrudeFeature* method to create Extrusion. You can refer to reference document for more informations. In this example input parameter are *("Extrude1", spBaseSketch, 100, Blind, 0, Blind, false)*. As you can see, it uses seven parameters to create an extrusion feature. *“Extrude1”* is a name of the feature, *spBaseSketch* for reference, and next numeric parameter is for depth information. So that code extrude for size 100. And the information of the other parameters are in the reference document. Please refer to it

But important thing is that what is the input for reference. The 2nd variable *spBaseSketch* is the reference of the feature, that is, *sketch1* that we drew is extruded. After all we can get the L-shape solid model if we run the sample code.

Next stage is drawing a circle that is second sketch.

|  |
| --- |
| IReferencePtr spReference2 = spPart->SelectBrepByName("Extrude1,Sketch1,Line2,0,0,0,ExtrudeFeature:0,0:0;0");  IStdSketchFeaturePtr spSketch2 = spFeatures->AddNewSketchFeature(\_T("Sketch2"), spReference2);  spSketch2->SetCoordinateSystem(150.0, 100.0, 50.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0);  ISketchEditorPtr spEditor2 = spSketch2->OpenEditor();  spEditor2->Create2DCircleCenterPoint("Circle1", 0, 0, 15);  spEditor2->Close(); |

For drawing a circle, you have to choose a position for drawing. In CAD UI, you can choose a surface by mouse click to sketch. When you use API, it uses a little different method.

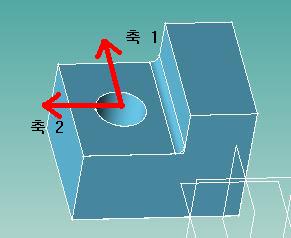
Mainly there are two methods that are Geometry and Topology methods. Geometry method is a surface selection method of geometrical condition. For example, if position (0, 0, 0) and axis vector (1, 0, 0), (0, 1, 0) are the information’s to insert, XYPlane is sketch plane. TransCAD also offers Geometry method. Recentry, IlwhanSong is working on it.

Second method, Topology method, is shown in example. In code, ("Extrude1,Sketch1,Line2,0,0,0,ExtrudeFeature:0,0:0;0") which is an insert variable for SelectBrepByName function is important part. This variable means “A plane which is generated by Extrude1 and Line2 of Sketch1.” Is Line2 a line which is connected between (200,100) and (100,100)? A plane is generated by Extrude1 is saved as ("Extrude1,Sketch1,Line2,0,0,0,ExtrudeFeature:0,0:0;0").

Recently, Face and Edge which are generated by any shape are named and saved in TransCAD. You can refer Dr. DuwhanMun’s paper for rules of naming. There is no problem for early stage of translator developing. For developers, Geometry method is more simple, Geometry method contains possibility of error. I can’t tell you which method is better.

After seleting face, make “spSketch2” object which is IStdSketchFeaturePtr using the “AddNewSketchFeature” function same as first sketch.

in the next step, use the “SetCoordinateSystem” function. This function defines a new axis system of any sketch. First each 3 variables are the x, y, z value of origin. And next 3 variables expresses the first axis, the next 3 variables expresses the second axis. So, like the picture, there is an axis system which has origin (150,100,50), Z axis for first axis, and X axis for second axis.



After this, it’s not different from previous steps. Open SketchEditor and sketch the circle whose radius is 15 on (0,0) using the “Create2DCircleCenterPoint” function. In previous steps, we already defined a new axis system, so the circle can be created like the picture.

you should excute codes by yourself and understand the meaning. The basic meaning is already explained. If a new function is needed, You can see all the informations about it in the API reference document.

This is the end of the document about an example of TransCAD.

1. **Inventor example(API)**

Next, the example code of L-block using inventor API will be described. If you use macro file, then you’d better skip this part. Inventor API’s structure is almost same as of the Automation API’s of TransCAD. Therefore, if you already checked the previous section: examples of TransCAD, you are aware of that code flow that is also same. One different thing is that inventor API is based on ‘visual basic’ and has more functions. Therefore, the code which will be described is made using ‘visual basic’ programming language. So, you’d better check basic grammar of ‘visual basic’, if you are a begginer of this language.

It can be possibe that the structure of inventor API is different from other CAD systems, basic flow is same as others. Let’s look at the example of inventor API’s code in oder to apply it to your CAD system. And it will be helpful when you make your own L-block in your CAD system.

|  |
| --- |
| Sub Make\_LBlock()  'Create New Par Document  Dim oPartDoc As PartDocument  Set oPartDoc = ThisApplication.Documents.Add(kPartDocumentObject, \_  ThisApplication.FileManager.GetTemplateFile(kPartDocumentObject))    Dim oCompDef As PartComponentDefinition  Set oCompDef = oPartDoc.ComponentDefinition  'Create Sketch Plane in XY Plane  Dim oSketch As PlanarSketch  Set oSketch = oCompDef.Sketches.Add(oCompDef.WorkPlanes.Item(3)) |

First, I made the function ‘Make\_Lblock()’. Because, the program what I want to make is that L-block appears immediately after I started the program, there is no input variables and form structure.

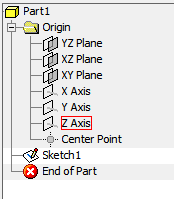
Inventor also has components in the order of Inventor 🡪 Document 🡪 Part Document 🡪 Part( Component Definition). Since document class’s funcions are not used often, here, we only set Part Document level and Part Composition Denfinition level’s variables.

Next, oSketch variable is declared, that is PlanarSketch type variable. It does the same job as IStdSketchFeaturePtr in TransCAD. Look at the following line.

Set oSketch = oCompDef.Sketches.Add(oCompDef.WorkPlanes.Item(3))

First, oCompDef variable controls part level of Inventor, which is equivalent to spPart in TransCAD. oCompDef.sketches. Add function adds new sketch to currently instantiated document. Add function has one input variable and one additional input variable, and this is further explained in reference guide. First input variable should be PlanarEntity type object, which decides the plane where to put the sketch. In above example code, oCompDef.WorkPlanes.Item(3) is given for that.

3 WorkPlane is given as default in Inventor. WorkPlane’s first item, WorkPlanes.Item(1) is YZ Plane, second item is XZ Plane, and the third item is XY Plane. Exactly same as the picture below.



Additionally, 3 WorkAxis and one WorkPoint are given too.

We gave WorkPlane’s third item as Add function’s input variables. This means that we set our sketch plane as XY plane. Now we start the sketch.

ThisApplication was referred in front of this guide. It refers the highest class, which is Inventor program. Subordinate class, TransientGeometry is a class that includes functions that generates basic geometry. We can make point, line, surface, curve, and vector etc. We used it to make point in this case.

With ThisApplication.TransientGeometry

Dim oLines(1 To 6) As SketchLine

Set oLines(1) = oSketch.SketchLines.AddByTwoPoints(.CreatePoint2d(0, 0), .CreatePoint2d(100, 0))

Set oLines(2) = oSketch.SketchLines.AddByTwoPoints(oLines(1).EndSketchPoint, .CreatePoint2d(100, 50))

Set oLines(3) = oSketch.SketchLines.AddByTwoPoints(oLines(2).EndSketchPoint, .CreatePoint2d(40, 50))

Set oLines(4) = oSketch.SketchLines.AddByTwoPoints(oLines(3).EndSketchPoint, .CreatePoint2d(40, 80))

Set oLines(5) = oSketch.SketchLines.AddByTwoPoints(oLines(4).EndSketchPoint, .CreatePoint2d(0, 80))

Set oLines(6) = oSketch.SketchLines.AddByTwoPoints(oLines(5).EndSketchPoint, oLines(1).StartSketchPoint)

End With

The second line uses matrix variable to declare SketchLine type variables. We need 6 lines for creating L-Block.

From 3rd line to 8th line, commands are given for sketching 6 lines. Let’s look at them one by one. First, oSketch is XY plane. oSketch(PlanarSketch class) has many subordinate entities. Since we need lines, we go into SketchLines class. Then, we can use two points as input and draw a line that connects those points by using AddByTwoPoints function. (Carefully look at the Inventor API Reference document. Visual Basic provides Object browser for you)

Therefore, if you put lines in each component of oLines matrix. Line that connect (0,0) and (100,0) is assigned to oLine(1), line between (100,0) and (100,50) is assigned to oLine(2) and so forth.

If you try to draw the second line by directly connecting (100,0) and (100,50) instead of using the method shown above (not using oLines(1)), it will not be drawn successfully. You should set the restriction separately. If you want to learn more, try to find the method for giving restriction.

Now we have L shape sketch in our XY plane.

Next, we will extrude this sketch.

Dim oExtrude\_1 As ExtrudeFeature

'Profile for first Extrude

Dim oProfile As Profile

Set oProfile = oSketch.Profiles.AddForSolid

'First Extrude

Set oExtrude\_1 = oCompDef.Features.ExtrudeFeatures.AddByDistanceExtent(oProfile, 80, kPositiveExtentDirection, kJoinOperation)

First, we declared oExtrude\_1 variable as ExtrudeFeature type. We can use this variable when we want to change the extrude depth parameter later on after the extrusion of the sketch is done.

oProfile variable is declared as Profile type, which is equivalent of the reference in TransCAD. To easily understand this, run the code before this level and press Extrude button in Inventor user interface. There are two kinds of Profile, one is extrusion of boundary line of the sketch(AddForSurface) and the other is extrusion of inside surface of the sketch(AddForSolid). Now we use AddForSolid by accessing Profile of oSketch and calling AddForSolid function. We save this in oProfile variable.

Next, we save the extrusion into oExtrude\_1. Similar to TransCAD, we call AddByDistance function by accessing ExtrudeFeature inside the oCompDef’s Feature class. Input variables can be decided by looking at the reference guide document of Inventor API. In our case, we chose Profile that we saved previously, extrude distance, direction, and Joint Operation for the input.

Additionally speaking, in above code, you can only write the line below,

oCompDef.Features.ExtrudeFeatures.AddByDistanceExtent(oProfile, 80, kPositiveExtentDirection, kJoinOperation)

to execute extrusion in Inventor. The reason of saving this into oExtrude\_1 is to allow easy access to the face and edge later created by the extrusion.

The next and last part is to sketch the circle.

Dim oPoint As Point

Set oPoint = ThisApplication.TransientGeometry.CreatePoint(70, 50, 40)

Dim oWorkPoint As WorkPoint

Set oWorkPoint = oCompDef.WorkPoints.AddFixed(oPoint)

Set oSketch = oCompDef.Sketches.Add(oExtrude\_1.SideFaces.Item(4))

oSketch.OriginPoint = oWorkPoint

'Create Circle

With ThisApplication.TransientGeometry

Call oSketch.SketchCircles.AddByCenterRadius(.CreatePoint2d(0, 0), 15)

End With

As same as TransCAD, you should first set the coordinate that has origin at the center of the circle that will be drawn. Inventor provides WorkPoint and it is (0,0,0). To select the coordinate with other origin, you should add WorkPoint first. CreatPoint function which is inside the TransientGeometry class can be called to create point at (70,50,40). Then we created oWorkPoint variable as WorkPoint.(oCompDef.WorkPoints.AddFixed(oPoint))

Then we added new sketch by using oCompDef.Sketches.Add. Input variable is oExtrude\_1.SideFaces.Item(4) in this case. To select the plane for sketching the circle, we access to the fourth plane of our previous extrusion by using oExtrude\_1.

Then, we access to OriginPoint where oSketch’s origin is saved and changed the origin into WorkPoint that we set as (70,50,40) previously. Now we can call AddByCenterRadius function of SketchCircle class inside oSketch, and draw circle by giving input of origin and radius.

The explanation about Inventor’s API is up to this point. Basically, Inventor has similar structure as TransCAD API; therefore, not much difference was found. However, users with different CAD system’s API will find more difference from their cases.

It would be much helpful to look at the reference guide of API that you are using.

1. **SolidWorks Example(Macro)**

SolidWorks is one of the CAD systems that uses Macro. Macro is ususally saved in text form, and the CAD system can save and read this kind of text file. In order to save Macro file, we should start with “Macro file save mode” through specific command and draw a part mannually, then a Macro file is created corresponding to the part drawn (depending on CAD system). By reloading this Macro file in CAD system later, the same part that is drawn mannually can be drawn automatically by CAD system.

Similar to the example of Inventor API, the command lines and methods of using Macro can be different depending on which CAD system is used. Current example is just for reference; therefore, method for each CAD system should be learned by yourself in details. However, similar structure does exists, so we recommend you to read this guide.

In this chapter, we will explain about Macro in details with an example of Macro file of L-block in SolidWorks.

Dim swApp As Object

Dim Part As Object

Dim SelMgr As Object

Dim boolstatus As Boolean

Dim longstatus As Long, longwarnings As Long

Dim Feature As Object

Sub main()

Set swApp = Application.SldWorks

Set Part = swApp.ActiveDoc

Set SelMgr = Part.SelectionManager

swApp.ActiveDoc.ActiveView.FrameState = 1

The code starts with declaration of variables. Looking at commands such as Dim and Set, we noticed that SolidWorks uses Macro file based on VB(Visual Basic) language. Here, the most important variables are swApp, Part, SelMgr, Feature. Deducing from the name of each variable, we already learnd that swApp controls SolidWorks CAD systems, Part is responsible for each part, Feature is responsible for each feature, and SelMgr manages “select”(reference select) instance.

Sub main() represents main function of Visual Basic. This function could be understood if you have background knowledge about Visual Basic (CATIA Macro also uses VB).

In the next line, swApp variable is set as SldWorks of Application. This statement assigns SolidWorks CAD program of local computer to swApp variable. Variable, Part, is declared as ActiveDoc of swApp. This means the active document of swApp is assigned to variable, Part. SelMgr is declared as SelectionManager of Part.

In fact, this part is not important to developer since this part of code does not change regardless of which part you desgin. Especially, in CAD system that uses Macro, this part is less important because users command CAD systems to start “Macro file save mode” before they start drawing the part, and the Macro file is automatically generated by the CAD system.

From now on, we will explain in earnest about creating L-Block.

boolstatus = Part.Extension.SelectByID2("정면", "PLANE", 0, 0, 0, False, 0, Nothing, 0)

Part.InsertSketch2 True

Part.CreateLine2 0, 0, 0, 0.01, 0, 0

Part.CreateLine2 0.01, 0, 0, 0.01, 0.006, 0

Part.CreateLine2 0.01, 0.006, 0, 0.003, 0.006, 0

Part.CreateLine2 0.003, 0.006, 0, 0.003, 0.012, 0

Part.CreateLine2 0.003, 0.012, 0, 0, 0.012, 0

Part.CreateLine2 0, 0.012, 0, 0, 0, 0

Part.ClearSelection2 True

Part.InsertSketch2 True

Variable named, “boolstatus”, is instantiated already in the beginning of the code. This variable is used for the allocation of “true” or “false” value which will be returned from other functions.

The called function, SelectByID2 is a member fuction of Extension class that is subordinate of Part class. As its name defines, the function selects specific geometry through ID. There were similar functions in TransCAD and Inventor explained before. This function is called to select XY plane where the sketch will be drawn. The input parameters “정면”(front), and “Plane”, etc are entered. For the detail explanation of this function please look at the API guide document of SolidWorks. To be short, we can notice that this function is just selected front plane.

The number 2 at the end of the function name is for the compatibility of Macro file between different versions. In the previous version, there was function named, SelectByID, in Macro file. As the function is updated in the new version, the function name should be distinguished from its old name in previous version. This is to prevent possible errors coming from different operation methods or different input parameters of the new version.

Therefore, when the vendor wants to update the function in new version, they can change name of the function into SelectByID3 or SelectByID4 later on, so that they can make the file operate properly without error.

Back to the code, the next line shows that InsertSketch2 function is set as True. This means that the sketch will be drawn from this point. Next lines will be easily understood, and after finishing the line sketches, ClearSelection2 is called and then InsertSketch2 is called.

As now, L shape sketch is finished and we will move on to the next part.

boolstatus = Part.Extension.SelectByID2("선1@스케치1", "EXTSKETCHSEGMENT", 0.005, 0, 0, False, 0, Nothing, 0)

Part.FeatureManager.FeatureExtrusion2 True, False, False, 0, 0, 0.008, 0, False, False, False, False, 0, 0, False, False, False, False, 1, 1, 1, 0, 0, False

Part.SelectionManager.EnableContourSelection = 0

Again, SelectByID2 is used to select something. Input parameters are “선1@스케치1”(line1@sketch1) and “EXTSKETCHSEGMENT”. The first segment line of recent sketch is selected. 0.005 denotes x coordinate.

After the selection, extrusion should be happened. Extrude feature is created by calling the function, FeatureExtrusion2, in FeatureManager class that is subordinate class of Part class. Then EnableContourSelectionis set as false.

boolstatus = Part.Extension.SelectByID2("", "FACE", 0.0065, 0.006, 0.004, False, 0, Nothing, 0)

Part.InsertSketch2 True

Part.ClearSelection2 True

Part.CreateCircleByRadius2 -0.0065, 0.004, 0, 0.001

Part.ClearSelection2 True

Next part is about circle sketch. In SolidWorks, the face created by extrusion does not have a name. Instead of selecting name, point(0.0065,0.006,0.004) is entered as input in the function. It is clarified that this selection is face with input “FACE”. Afterward, InsertSketch2 and ClearSelection2 functions are used as similar as L shape sketch, and by using CreateCircleByRadius2, circle is drawn.

In SolidWorks, local coordinates cannot be used, so in order to draw a circle absolute coordinate (-0.0065,0.004,0) is used for the center. In Inventor, center point is set first, and then local coordinate (0,0) is used for the center of the circle.

This sort of difference among CAD systems should be analyzed in order to understand CAD systems and develop the translator because the translator is developed for the exchange of data among different CAD systems.

Next part is the last part. SolidWorks macro is simple so it requires relatively short explanation.

boolstatus = Part.Extension.SelectByID2("Arc1", "SKETCHSEGMENT", 0, 0, 0, False, 0, Nothing, 0)

Part.FeatureManager.FeatureCut True, False, False, 0, 0, 0.002, 0, False, False, False, False, 0, 0, False, False, False, False, 0, 1, 1

Part.ClearSelection2 True

boolstatus = Part.Extension.SelectByID2("", "EDGE", 0.003027566077776, 0.005978589108679, 0.005298091073996, False, 1, Nothing, 0)

Part.FeatureFillet5 195, 0.002, 0, 0, 0, 0, 0

End Sub

This also starts by using SelectByID2 function. This is before the operation of Cut Extrude of the circle, so we should select the previous sketch of the circle. Circle is classified as arc. SKETCHSEGMENT named “Arc1” is selected. The reason of using SKETCHSEGMENT instead of EXTSKETCHSEGMENT is not clear but since this Macro file does not have any problem so we will not go to details about this issue.

Next, as extrusion we did before, we call FeatureCut function in FeatureManager class, and we create cut extruction on the part.

Lastly, we use SelectByID2 function to select unnamed EDGE by using point (0.003027566077776, 0.005978589108679, 0.005298091073996) and call FeatureFillet5 function to create fillet feature on the part.

As you can see till now, Macro file contains all the designing process of the part that is designed by user manually (Although this is not same as Macro files of different CAD systems). Different from Inventor or TransCAD, the Macro file of SolidWorks does not work in a way of saving the reference of selected plane or line and passing that information to the Feature. Therefore, you should think about how to manage this part during translation.

1. **Translation Method**

Previous exmaples were about drawing Part in each CAD systems using Macro or API. To implement the translator the ability to read before the drawing is required. In other words, to translate from A to B, the translator should first read from system A before drawing in system B.

In this chapter, two simple example codes will be studied to learn about overall translator’s process and main problems should be solved.

Public Sub INV\_PRE\_Feature(oFeature As PartFeature)

Dim oSketchEntity As SketchEntity

Select Case oFeature.Type

Case kExtrudeFeatureObject

Dim oExtrudeFeature As ExtrudeFeature

Set oExtrudeFeature = oFeature

INV\_PRE\_ExtrudeFeature oExtrudeFeature

Case kHoleFeatureObject

Dim oHoleFeature As HoleFeature

Set oHoleFeature = oFeature

INV\_PRE\_HoleFeature oHoleFeature

Case kFilletFeatureObject

Dim oFilletFeature As FilletFeature

Set oFilletFeature = oFeature

INV\_PRE\_FilletFeature oFilletFeature

Case kChamferFeatureObject

Dim oChamferFeature As ChamferFeature

Set oChamferFeature = oFeature

INV\_PRE\_ChamferFeature oChamferFeature

Case kSweepFeatureObject

Dim oSweepFeature As SweepFeature

Set oSweepFeature = oFeature

INV\_PRE\_SweepFeature oSweepFeature

Case kRevolveFeatureObject

Dim oRevolveFeature As RevolveFeature

Set oRevolveFeature = oFeature

INV\_PRE\_RevolveFeature oRevolveFeature

Case Else

End Select

End Sub

Code above shows the basic outline for translating Inventor part into TransCAD part.

Macro-Parametric method should preserve the part design history. Accordingly, the translation should be done in the order of original part’s design history. In consideration of the order of steps in previous L-Block’s design,

Select XY Plane 🡪 L shape sketch 🡪 Extrude Feature created

Select plane 🡪 circle sketch 🡪 Cut Extrude Feature created

Select Edge 🡪 Fillet Feature created

these three steps are used. Therefore, to translate this part, these three features(Extrude, Cut Extrude, Fillet) should be reconstructed in TransCAD.

In the code, oFeature is received as input and its type is determined. For each type of feature, corresponding translating function is called. Consequently, for L-Block, features are successively received as ExtrudeFeature, ExtrudeFeature, FilletFeature.

Theoretically, the next step is just calling functions that draw these features of TransCAD in this sequence. However, there is a little hardship.

Let’s consider about the translation of the first step,

Select XY Plane 🡪 L shape sketch 🡪 Extrude Feature created

of the three steps mentioned above.

For the translation, steps below should be followed..

**Check Extrude Feature(From Inventor) 🡪 Check the sketch used for Extrude Feature 🡪 Check Reference of the Sketch 🡪 Check Sketch’s Entity(Point, Line, Arc etc) 🡪 Sketch’s Reference Generated(in TransCAD) 🡪 Sketch’s Each Entity Generated 🡪 Select the Sketch that will be used for Extrude Feature 🡪 Extrusion Generated**

Therefore, function: INV\_PRE\_ExtrudeFeature in the code executes rest of the steps except checking Extrude Feature. To check the sketch reference and the sketch that is used for feature are redundant tasks required for each feature, so it is better to make general function for this task and call when it is needed.

However, since each feature and sketch has different characteristics, it is hard to make general functions that can be used for all kinds of models. Yet, it is still important to make these function to have general purpose as much as possible, or we should develop a new translator for each model, if so, it becomes meaningless to make translator.

The tool that translates TransCAD model into Inventor model is not that much different.

Dim oTransFeature As TransCAD.IFeature

For Each oTransFeature In oTransFeatures

Select Case oTransFeature.Type

Case StdDefaultDatumPlaneFeature:

Case StdSketchFeature:

Call GetSketchInfo(oTransFeature)

Case StdSolidCutExtrudeFeature:

Call GetSolidCutExtrudeInfo(oTransFeature)

Case StdSolidFilletConstantFeature:

Call GetFilletConstantInfo(oTransFeature)

Case StdSolidProtrusionExtrudeFeature:

Call GetSolidProTrusionExtrudeInfo(oTransFeature)

End Select

Next

Here again, features are distinguished and checked first, and sequentially, functions are called to generate these sketches and features in Inventor.

In 2010, Hybrid Naming method(implemented by Dr. Song) and general CAD system’s two Naming methods(Geometry-based, Topology-based) are important in process of determining reference during translation.

In L-Block translation, selecting plane for circle sketch can be easily done if input from user (use of mouse click) is available, whereas making this process automatic faces many obstacles. This will be commonly experienced by every developer.

The method of implementation not only depends on developer’s skill, but also depends on the direction of the project research.

1. **UG Example (Macro)**

Now, Translation using Macro file of Unigraphics NX will be described. A macro file is the history of modeling commands issued by a designer. The macro file or the log file corresponds to the procedural model. Most commercial CAD systems support macro file formats. Formats of macro files are different for each CAD system. Macro files of UG are text files at the level of graphical user interface (GUI).

We drew a L – Block in UG after switching on the Record macro option. The saved macro file was text file which looked like this -

NX 7.5.0.32

Macro File: D:\iCAD Lab\UG Macro files\L Block 1.macro

Macro Version 7.50

Macro List Language and Codeset: english 17

Created by Fahim on Mon Jan 02 13:46:27 2012

Part Name Display Style: $FILENAME

Selection Parameters 1 2 0.305441 1

Display Parameters 1.000000 21.628240 10.416667 -1.000000 -0.481623 1.000000 0.481623

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RESET

MENU, 1, UG\_SKETCH\_LINE UG\_GATEWAY\_MAIN\_MENUBAR ! <>

DIALOG\_BEGIN "Line" 0 ! DA2

BEG\_ITEM 1 (1 BOOL 0) = 0 ! Constraint Lock

DIALOG\_BEGIN "Persistent Dialog" 102001 ! Persistent

BEG\_ITEM 1 (1 BOOL 102001) = 1 ! Line

BEG\_ITEM 2 (1 BOOL 102001) = 0 ! Arc

BEG\_ITEM 5 (1 BOOL 102001) = 1 ! Arc by 3 Points

BEG\_ITEM 6 (1 BOOL 102001) = 0 ! Arc by Center and Endpoints

BEG\_ITEM 9 (1 BOOL 102001) = 1 ! Circle by Center and Diameter

BEG\_ITEM 10 (1 BOOL 102001) = 0 ! Circle by 3 Points

BEG\_ITEM 13 (1 BOOL 102001) = 1 ! Coordinate Mode

BEG\_ITEM 14 (1 BOOL 102001) = 0 ! Parameter Mode

BEG\_ITEM 17 (1 OPTM 102001) = 0 ! Segment Only

BEG\_ITEM 20 (1 BOOL 102001) = 0 ! Define Sketch Plane

DIALOG\_PERSISTENT\_END 102001

FOCUS CHANGE OUT 1

FOCUS CHANGE IN 1

FOCUS CHANGE IN 1

CURSOR\_EVENT 1001 106,0,201 ! motion\_pb, mb0/0+0, U\_Pre\_sel\_ND (T+:0+0)

CPOS 0.801056343765043,-1.60290151701686,0

CURSOR\_EVENT 1001 106,0,201 ! motion\_pb, mb0/0+0, U\_Pre\_sel\_ND (T+:0+0)

CPOS 0.534037562510028,0.267150252836126,0

CURSOR\_EVENT 1001 106,0,201 ! motion\_pb, mb0/0+0, U\_Pre\_sel\_ND (T+:0+0)

CPOS 0.267018781255014,0.267150252836126,0

CURSOR\_EVENT 1001 106,0,201 ! motion\_pb, mb0/0+0, U\_Pre\_sel\_ND (T+:0+0)

CPOS -2.30934860787248e-014,0.534300505672275,0

CURSOR\_EVENT 1001 106,0,201 ! motion\_pb, mb0/0+0, U\_Pre\_sel\_ND (T+:0+0)

CPOS -2.30934860787248e-014,0.534300505672275,0

CURSOR\_EVENT 1001 3,1,100 ! single\_pt, mb1/0+0, U\_Sel\_sngl (T+:0+0)

CPOS -2.30934860787248e-014,0.534300505672275,0

DIALOG\_BEGIN "Persistent Dialog" 102000 ! Persistent

BEG\_ITEM 0 (1 STRN 102000) = "8.03360410683027" ! XC

BEG\_ITEM 1 (1 STRN 102000) = "48.2011350359919" ! YC

BEG\_ITEM 2 (1 STRN 102000) = "359" ! Length

BEG\_ITEM 3 (1 STRN 102000) = "284" ! Angle

BEG\_ITEM 4 (1 STRN 102000) = "0" ! Radius

BEG\_ITEM 5 (1 STRN 102000) = "0" ! Sweep Angle

BEG\_ITEM 6 (1 STRN 102000) = "0" ! Diameter

BEG\_ITEM 7 (1 STRN 102000) = "0" ! Relative Angle

From the Macro file, we noticed that the macro file not only records the command history but also it records the screen coordinates that come from the entity selection by a pointing device. Again, we noticed that it records Undo/Redo history in addition to modification history and default settings in their macro files. It is very difficult to translate such macro files.

1. **UG Translator(API) Setup and Usage Guide**

In this chapter, different from the previous one, we will talk about actual installation of translator and usage guide rather than code.

Each Translator has different characteristics. In case of UG, implementing environment is very sensitive factor in terms of implementation; therefore, we will use Vmware(virtual machine tool) to sync the environment of implementation even if it will be done by different people. By using virtual machine, user can implement the translator in the same environment regardless of user’s PC environment(WinXP or Win7, 32bit or 64bit). Also the compatibility issue between different versions of CAD systems can also be resolved since everyone uses the same version of the system. Maybe we should update the version of the system inside the Vmware once in a while.

In Vmware we use UG NX version 7.5 and Window7 32bit for OS.

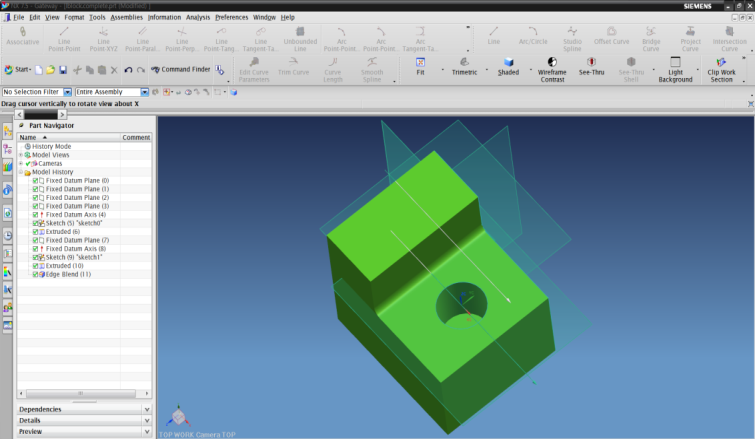


Figure 1 UG NX 7.5 Operation

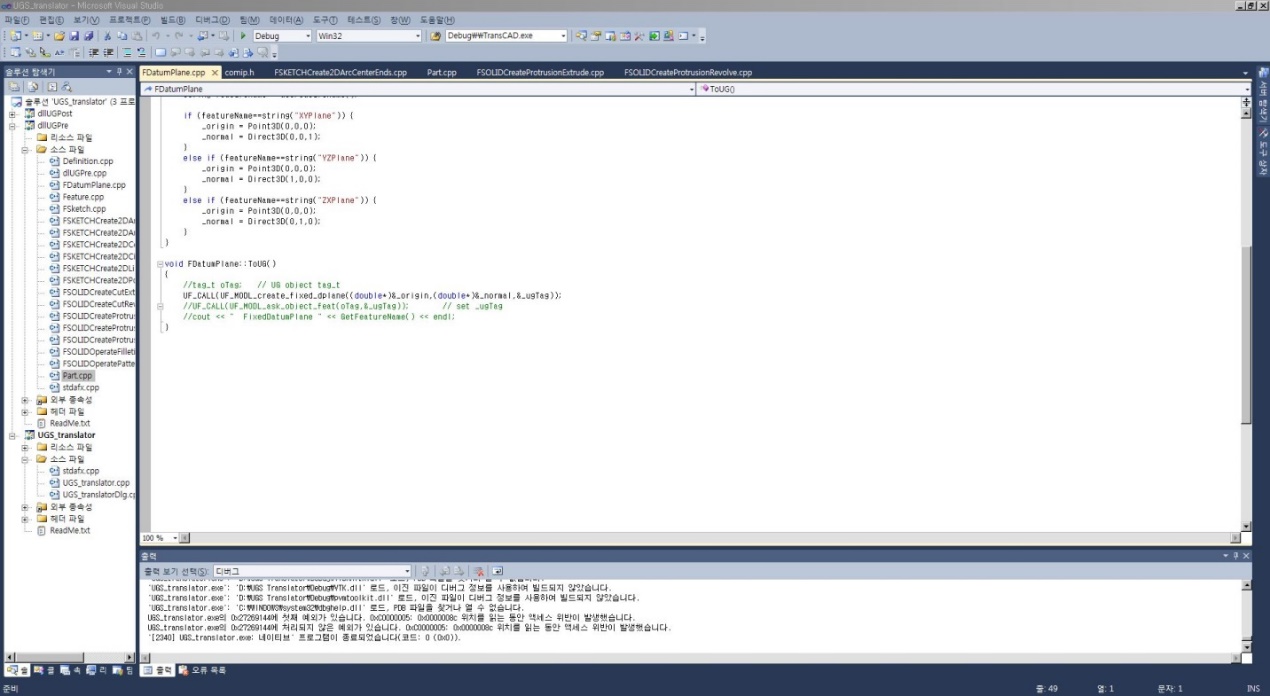
You can download free Vmwar player and run the image of virtual machine. It contains OS+Translator+CAD+Developing tool(Visual Studio).

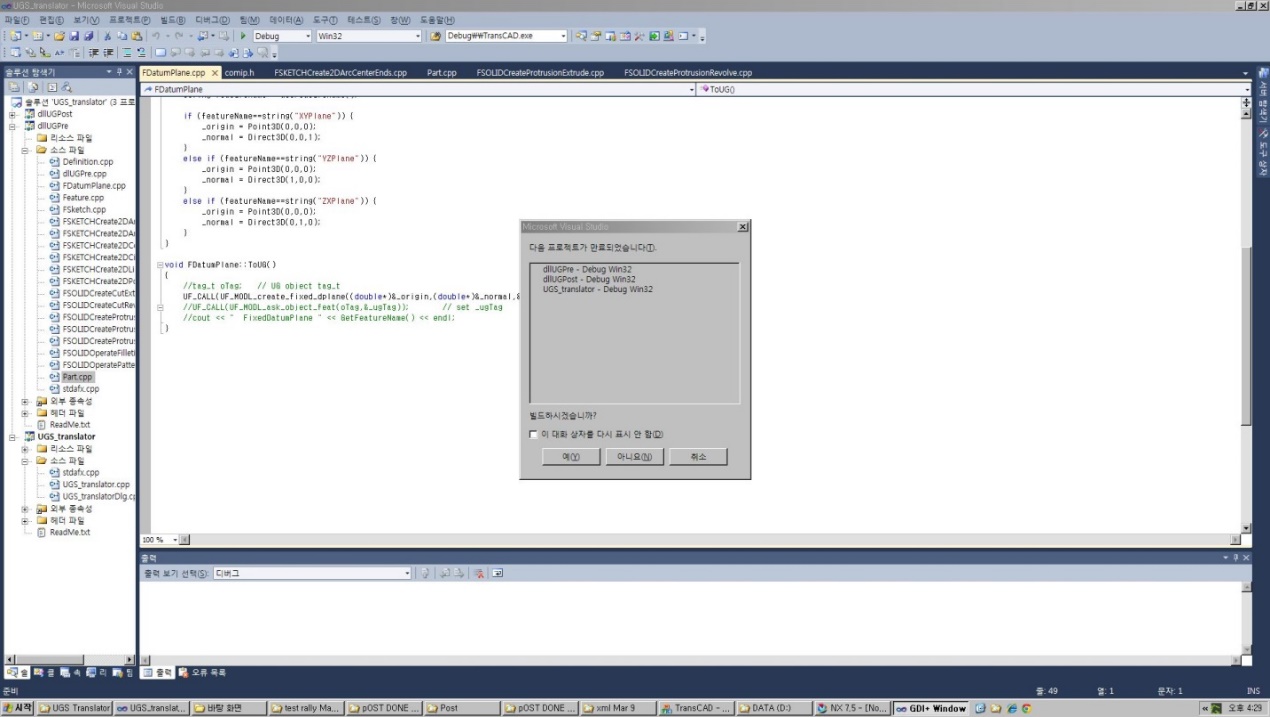
User opens the file with extension .vmx in VMwar Player by clicking “Open a Virtual Machine” command.

Inside the Vmware, you can follow the steps below to execute the Translator.

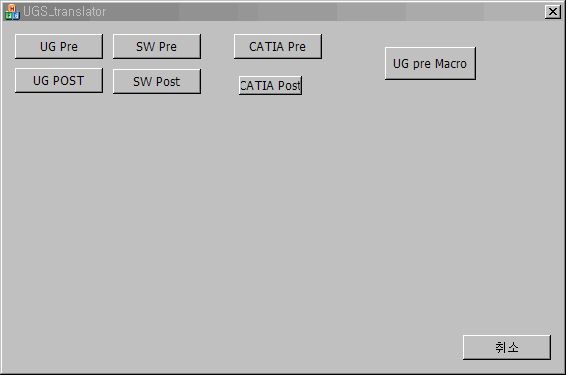
<UG Pre>

1. Click run/debug

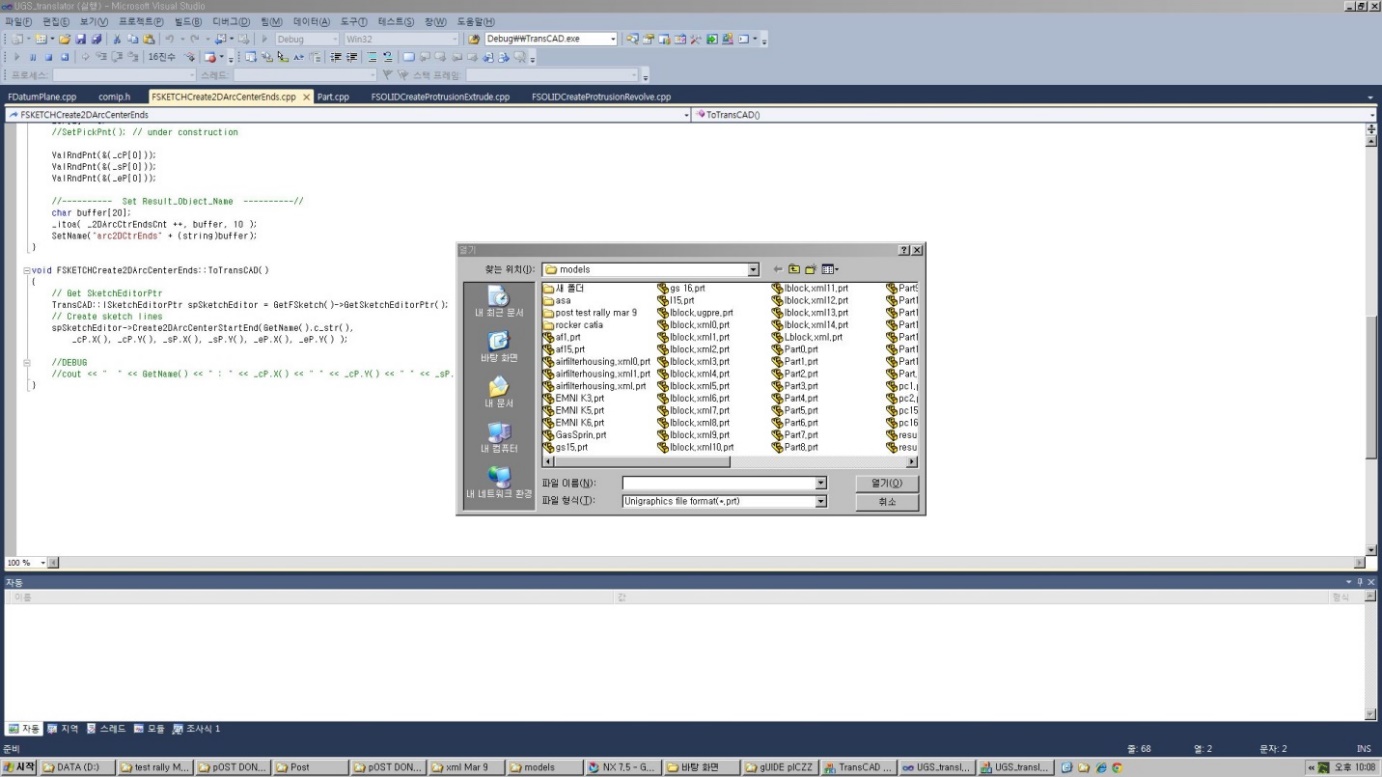




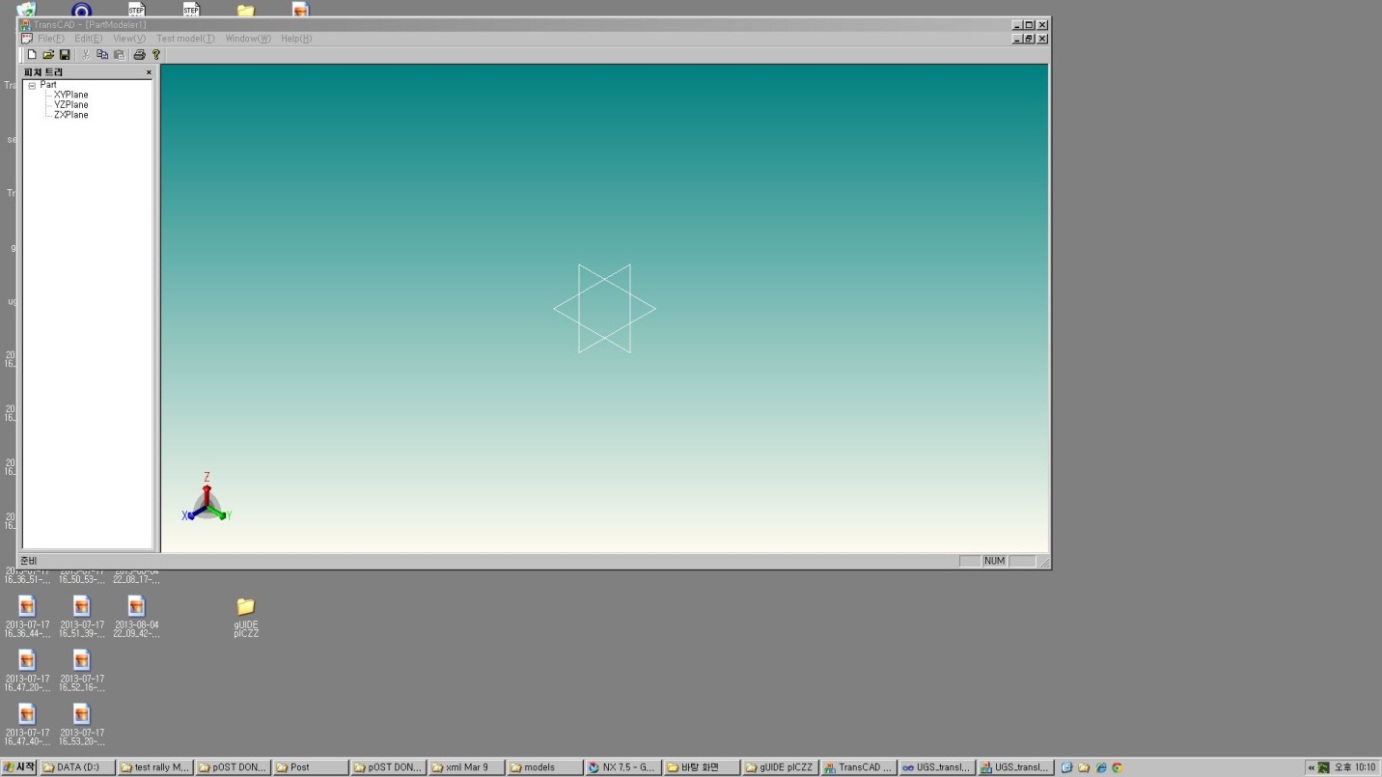
1. Click run/UG Pre



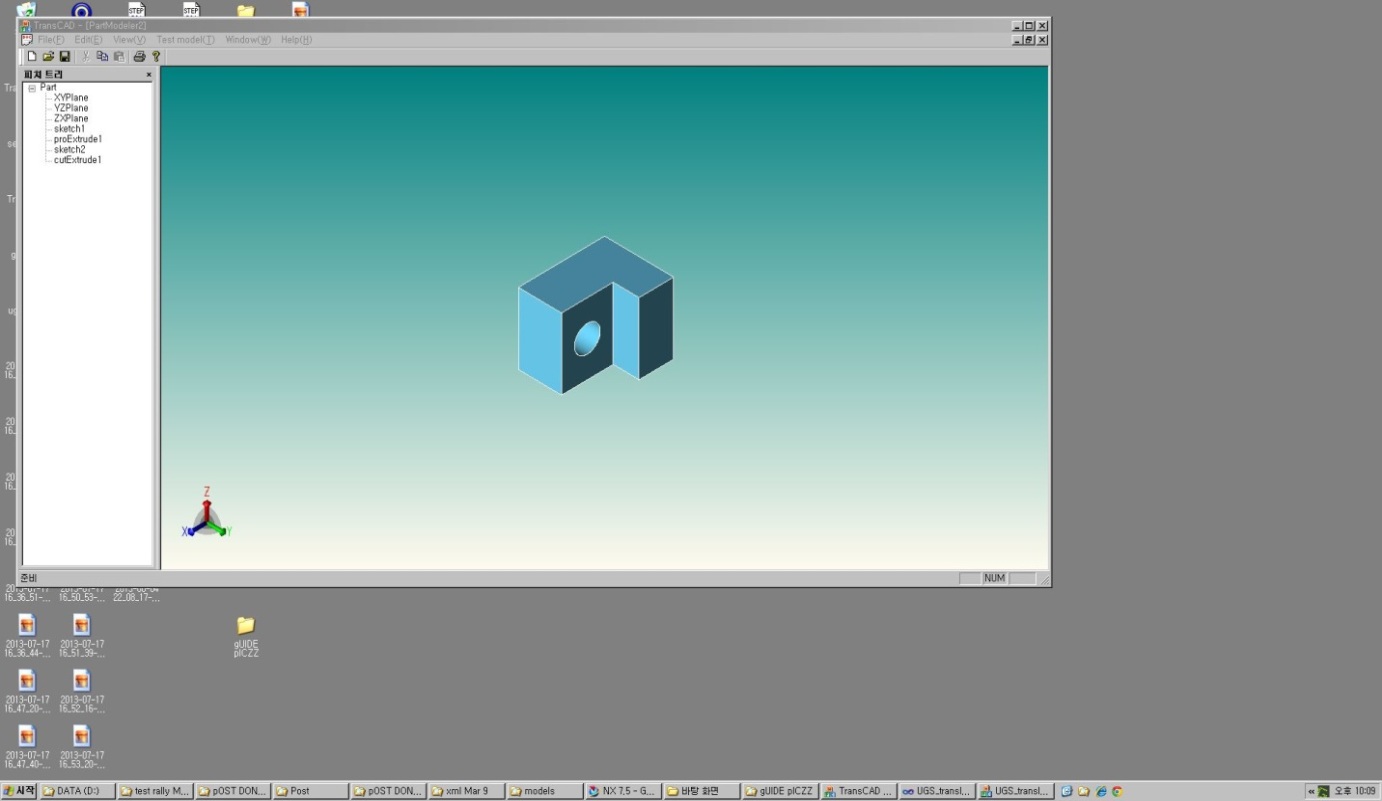
1. Open UG part file



Open ”LBlockUGPre.prt” file

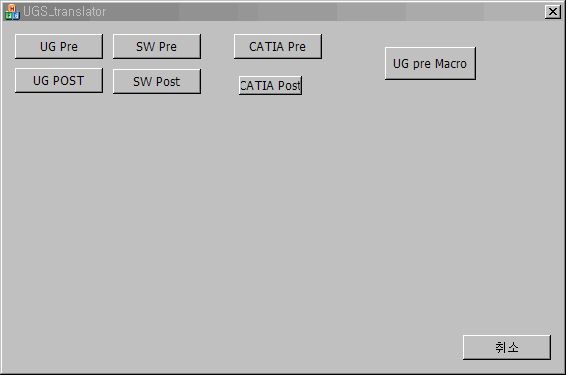


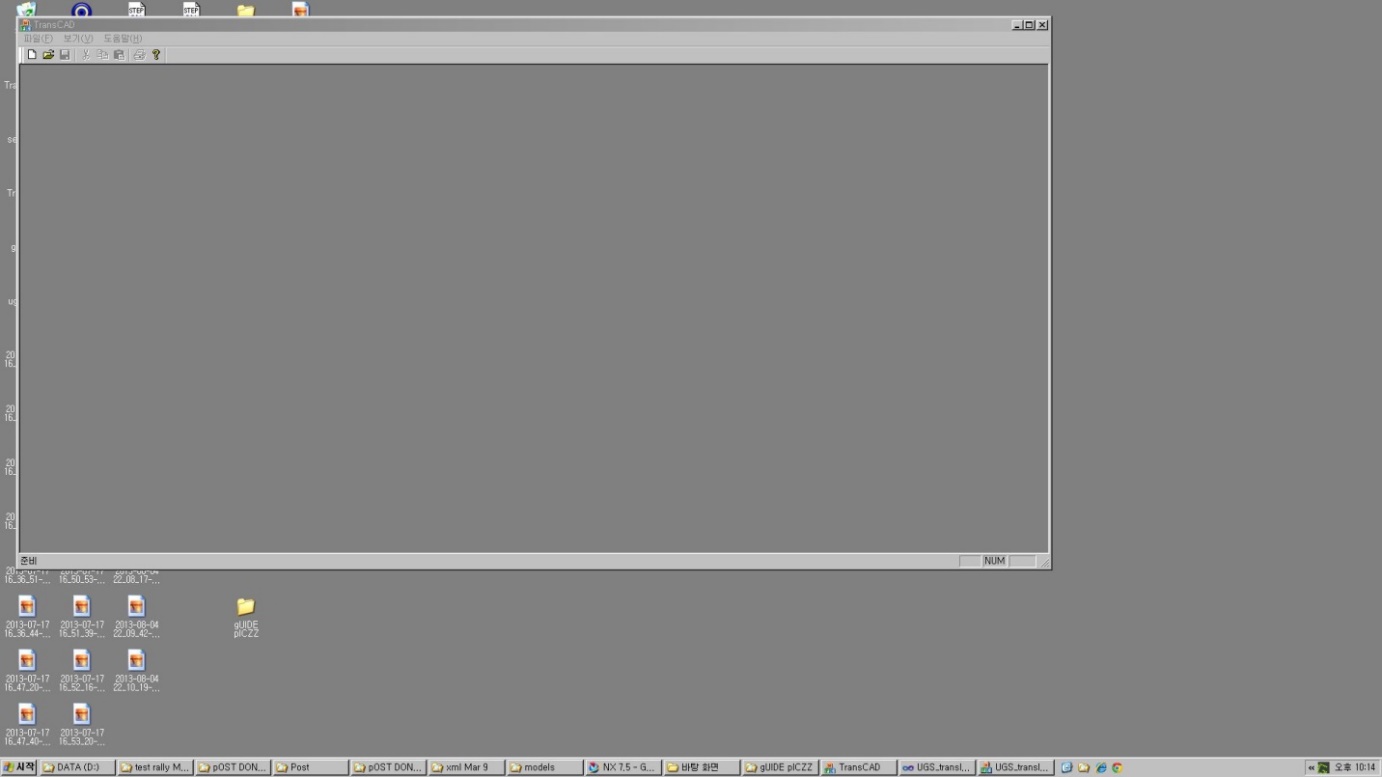
Before in TransCAD



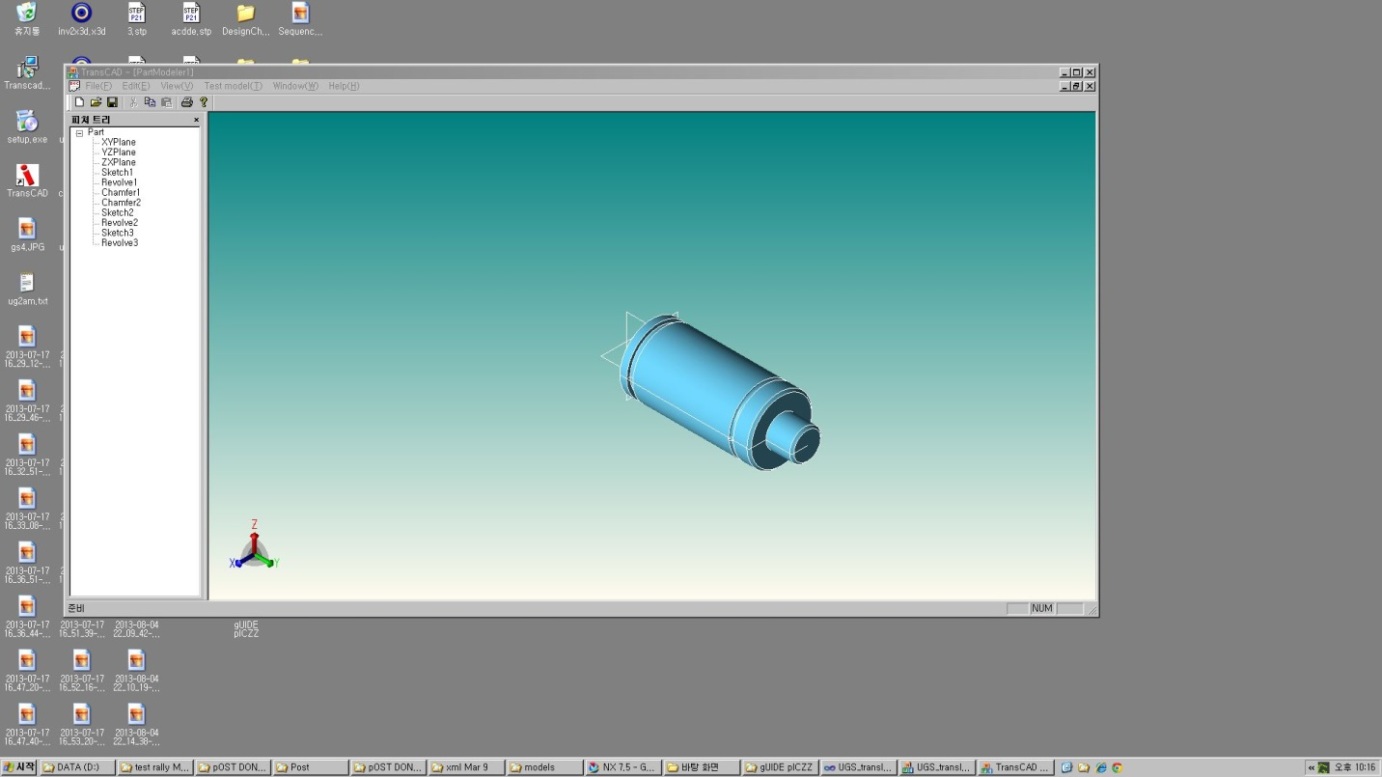
After in TransCAD

<UG Post>

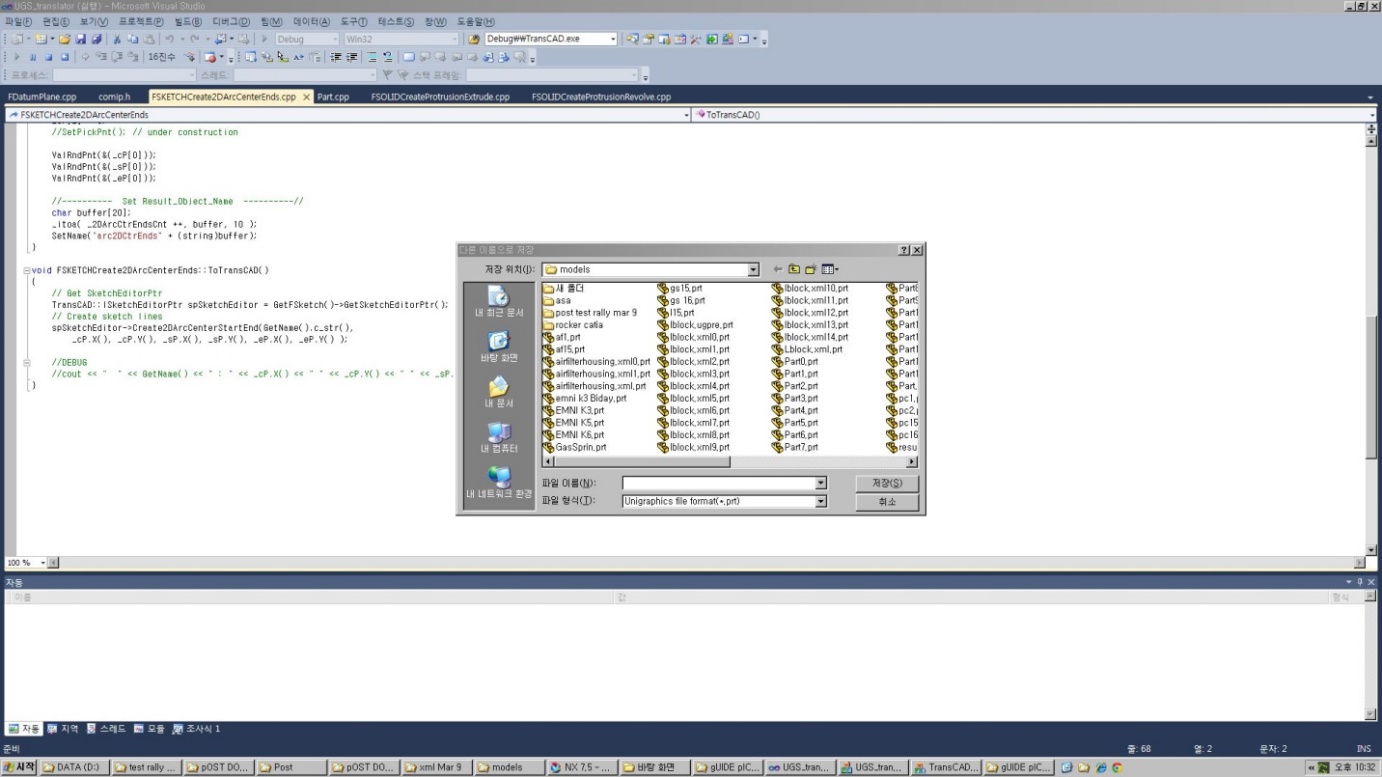




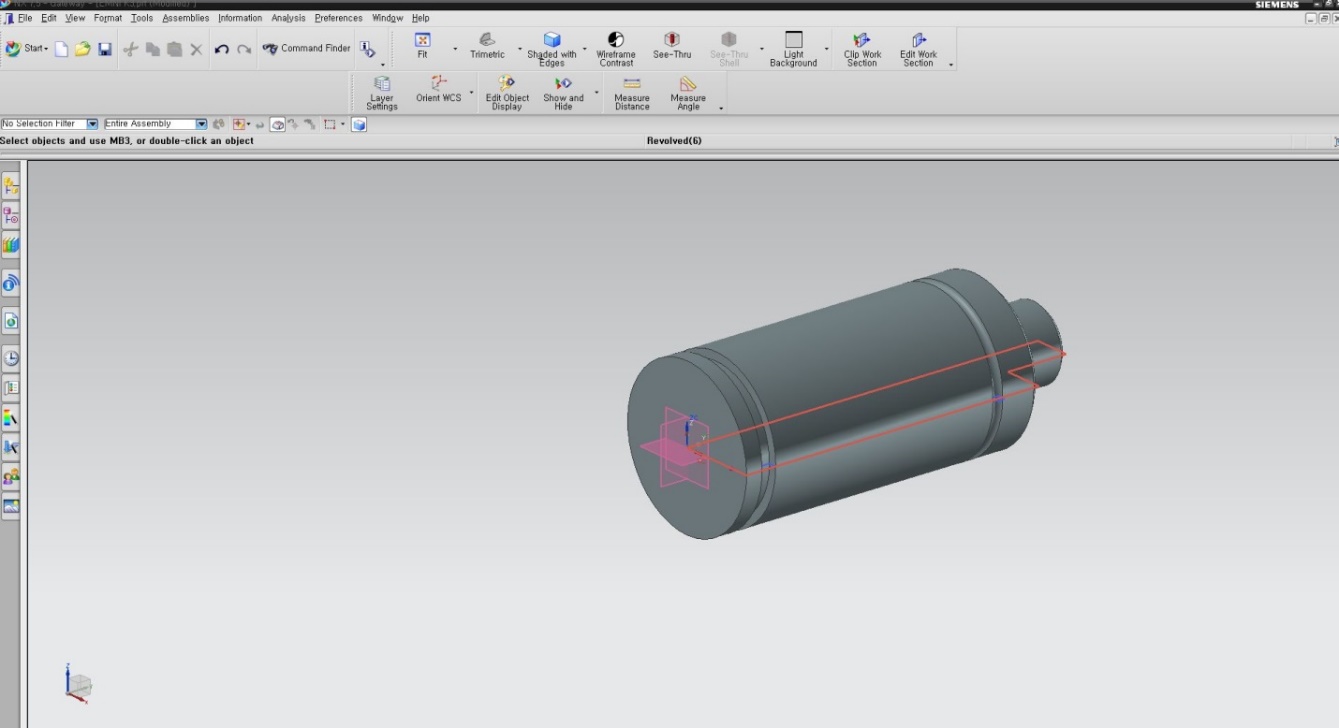
**TransCAD**



**Open Gas Spring in TransCAD**



**Save As GasSpringUGPost.prt**



**Open in UG NX**