**Implementation**

**Guidebook**

**iCAD**

**최종수정: 2014-03-18 이인환**

\*본 Implementation Guidebook의 담당은

연구실 신입생과 박사과정 선배가 한번씩 번갈아 가면서 맡습니다.

**<목차>**

1. 개요
2. TransCAD 예시.
3. Inventor 예시(API)
4. SolidWorks 예시(Macro)
5. 변환 방법
6. UG translator(API) 셋업 및 사용방법
7. 맺음
8. **개요**

이 문서는 Macro-parametric 프로젝트의 구현에 도움을 주기 위해 작성되었습니다.

TransCAD는 공통으로 사용되는 프로그램으로써, TransCAD자체에 대한 자세한 설명은 TransCAD Automation API Guideline, Automation API 참조 문서 그리고 Handbook에 담겨 있습니다. 본 문서에서는, 간단한 형상(주로 L-Block)을 각각의 CAD 시스템과 TransCAD에 표현하는 코드를 예제로, 실제 구현에 대해 도움을 주고자 합니다.

우리는 각자의 CAD시스템에서 TransCAD로, TransCAD에서 각자의 시스템으로 Part의 정보를 옮기는 프로그램을 구현해야 합니다. 그리고 각자의 CAD 시스템은 크게 나누어서 API를 사용하는 시스템(현재(2009.11) Inventor, Pro-E, UG)과 Macro를 사용하는 시스템(Solidworks, CATIA, UG)로 나눌 수 있습니다. 본 문서의 2장에서는 TransCAD에서 형상을 그리는 코드, 3장에서는 API를 통해 형상을 그리는 코드, 4장에서는 Macro를 통해 형상을 그리는 코드를 설명하겠습니다.

이 문서를 통해 구현에 많은 도움이 되길 바라며 본문을 시작하겠습니다.

1. **TransCAD 예시**

(전체 코드는 맨 뒤에 있습니다.)

위 코드는 TransCAD 시스템에서 L-Block을 그리기 위한 코드입니다. 이제 한줄 한줄 설명을 통해 위 코드를 파악해 보도록 하겠습니다.

using namespace TransCAD;

IDocumentsPtr spDocuments = m\_spApplication->Documents;

IPartDocumentPtr spDocument = spDocuments->AddPartDocument();

IPartPtr spPart = spDocument->Part;

우선 첫번째 부분입니다. 이 부분은 TransCAD에 새로운 파트를 그리기 위한 사전 작업을 하기 위한 부분입니다. 예를 들자면, 우리가 어떤 워드 문서를 작성하고 싶다고 한다면, 워드 프로그램을 켜고, 새 문서를 열어야겠지요. (TransCAD뿐만 아니라 다른 CAD 시스템에서도 비슷한 과정을 거쳐주어야 합니다.)

m\_spApplication은 TransCAD 사용을 위한 작업을 마쳤다면 정상적으로 선언이 되어 있을 것입니다. 위에서 보면 spDocuments라는 IDocumentsPtr을 m\_spApplication->Documents로 선언하였습니다. 다시 말하면, TransCAD(m\_spApplication이 TransCAD이지요.)의 Documents(새 문서 개념)를 spDocuments라고 설정을 해 준 것입니다. 이로써 TransCAD에 “새 문서”가 새로 생긴 것이죠.

다음 줄을 설명하기 위해선 CAD에 대한 이해가 조금 필요합니다. 각각의 CAD 시스템에 따라 다르겠지만, 기본적으로 CAD에 새 문서(편의상 앞으로 문서라고 부르겠습니다.)를 만들려고 하면, 그 문서가 무엇을 그리기 위한 문서인지 지정을 하게 됩니다. 간단한 예로 Part(파트 하나를 그리기 위한 문서)와 Assembly(파트들의 결합을 나타내기 위한 문서)가 있겠습니다. 그리고 Part를 선택했느냐, Assembly를 선택했느냐에 따라서 작업할 수 있는 내용이 달라지게 됩니다. 우리는 기본적으로 Part, 즉 부품 하나를 위한 도면을 기본 가정으로 생각하기 때문에 spDocuments->AddPartDocument();로 spDocument를 선언해주게 되는것입니다. Assembly로 새 문서를 만들고 싶을 때는 AddAssemblyDocument();를 선언해 주어야 하겠죠.

앞서 말한 워드 프로그램에서 새 문서의 예를 가지고 생각해 보면, 여기까지의 코드를 통해, 새 문서를 작업할 준비가 모두 되었다고 생각할 수 있습니다. 그러나, API를 통해 파트를 그리는 작업을 할 때에는 Part를 먼저 선언해 주어야 합니다. 즉, 그 다음줄에 있는 IPartPtr spPart = spDocument->Part; 부분인데요, 이 부분은 spPart라는 IPartPtr형 변수를, spDocument->Part로 선언해 주었습니다. 그 말은 앞으로 내가 그리게 될 Part가 무엇인지는 모르겠지만, 그것을 spPart라고 내가 지정하겠다는 것입니다. 이 코드를 통해, spPart 포인터를 가지고 우리는 우리가 그린 Part에 대해 접근할 수 있게 되는 것입니다.

이 부분이 잘 이해가 안되더라도 사실 큰 문제는 없습니다. 왜냐하면 대부분 상황에서 이 초기 선언부에 대한 설정을 우리가 건드릴 일이 없기 때문입니다^^

여기까지 마쳤다면, 기본적으로 새 문서를 생성하고, Part를 그릴 준비가 모두 되었다고 볼 수 있습니다. 그러면 그 다음으로 넘어가보죠.

이 부분이 앞으로 우리가 할 모든 작업의 기초가 되는 부분입니다. 가장 중요한 부분이지요.

IFeaturesPtr spFeatures = spPart->Features;

IReferencePtr spXYPlane = spPart->SelectObjectByName("XYPlane");

// Create a sketch on the sketch plane

IStdSketchFeaturePtr spSketch1 = spFeatures->AddNewSketchFeature(\_T("Sketch1"), spXYPlane);

// Open the sketch

ISketchEditorPtr spEditor1 = spSketch1->OpenEditor();

// Create sketch lines

spEditor1->Create2DLine2Points("Line1", 0.0, 0.0, 200.0, 0.0);

spEditor1->Create2DLine2Points("Line2", 200.0, 100.0, 100.0, 100.0);

spEditor1->Create2DLine2Points("Line3", 200.0, 0.0, 200.0, 100.0);

spEditor1->Create2DLine2Points("Line4", 100.0, 100.0, 100.0, 180.0);

spEditor1->Create2DLine2Points("Line5", 0.0, 180.0, 0.0, 0.0);

spEditor1->Create2DLine2Points("Line6", 100.0, 180.0, 0.0, 180.0);

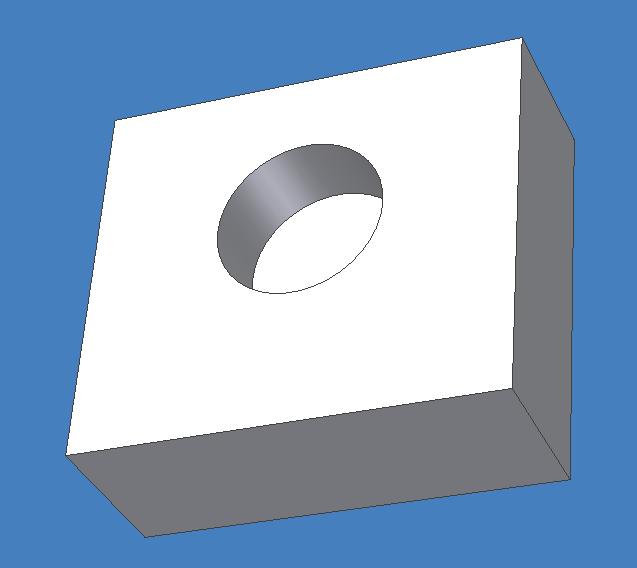
// Close the sketch

spEditor1->Close();

우선 여러분이 CAD 시스템을 조금은 만져 보았다고 가정하고 설명을 드리겠습니다. CAD에 무언가를 그려 보셨다면 아시겠지만 CAD 작업은 간단히 말하면 다음 세 가지 과정의 반복이라고 볼 수 있습니다.

Reference 선택 🡪 스케치 🡪 Feature 생성

간단한 예로, 정육면체에 구멍 하나가 뚫려있는 파트를 그린다고 생각해 봅시다. 다음과 같은 것 말이죠.



아주 간단합니다. 네모를 그리고, Extrude를 하고, 그로 인해 생긴 정육면체의 한 면을 선택하여 원을 그리고, Cut Extrude를 하면 되겠죠.

이 과정을 앞서 말한 세 단계에 맞추어서 설명해 보면

(reference 선택) 🡪 네모 스케치 🡪 extrude feature 생성 🡪 면 선택 🡪 원 스케치 🡪 cut extrude feature 생성

이와 같이 됩니다.

그러면 본론으로 들어가서, 코드의 첫 줄에서는 spPart->Features를 IFeaturesPtr형인 spFeatures라는 변수에 지정합니다. 이 spFeatures는 spPart 내의 Features의 정보를 담고있는 포인터입니다. 앞서, 아무것도 그리지 않았음에도 spPart를 먼저 지정해 준 것처럼, 이 부분도, 우리가 앞으로 Feature를 생성해 내기 위해 필요한 사전 작업을 한 것입니다.

다음 줄은 spXYPlane를 spPart->SelectObjectByName(“XYPlane”)라고 지정해 주었죠? 그리고 spXYPlane는 어떤 형인가요? IReferencePtr입니다. 즉 세 과정중 첫번째인 Reference 선택 부분이 이 부분입니다. 차근차근 알아봅시다.

우리가 처음 spPart라는, 아무것도 없는 객체를 생성했을 때, 사실은 TransCAD 내에서 몇 가지가 이미 만들어져 있습니다. 그 중 하나가 기본 reference plane 이지요. 즉, 첫 스케치를 그리기 위한 도화지가 되겠습니다. TransCAD 내의 세 가지 기본 reference plane의 이름은 “XYPlane”, “YZPlane”, “ZXPlane” 입니다.(TransCAD.EXE를 실행시켜 TransCAD를 띄워보면, 왼쪽에 나옵니다.) 이 세 개의 reference plane중 하나인 “XYPlane”을 SelectObjectByName 함수를 이용해서 지정한 것입니다.

다음 줄은 IStdSketchFeaturePtr spSketch1 = spFeatures->AddNewSketchFeature(\_T(“Sketch1”,spXYPlane) 입니다.

|  |
| --- |
| AddNewSketchFeature |
| 새 SketchFeature 객체를 만들고, 이를 추가합니다.   |  | | --- | | HRESULT AddNewSketchFeature([in] BSTR name, [in] IReference\* pSketchPlane, [out,retval] IStdSketchFeature\*\* ppVal); |   매개 변수  pSketchPlane  스케치가 그려지는 평면입니다.  ppVal  새 SketchFeature 객체입니다. |

우선 AddNewSketchFeature함수가 무엇을 하는지 알아보죠. API 참조 문서에 따르면, IFeature 클래스의 AddNewSketchFeature 함수는, 새 SketchFeature 객체를 만들고, 이를 추가 한다고 되어있습니다. 그리고 입력 변수로, name과 sketch plane을 받게 되죠(out으로 SketchFeature 객체를 보낼 수도 있지만, 이 부분은 필요하지 않으므로 생략하겠습니다.

실제로 쓰여진 코드에서, 우리는 새로 만들어질 sketch feature를, 이름은 “Sketch1”으로, 스케치가 그려질 평면은 spXYPlane, 즉 아까 우리가 reference 객체로 선언해준 XYPlane이 입력 변수로 들어갔습니다.

바로 다음 줄은, ISketchEditorPtr spEditor1 = spSketch1->OpenEditor(); 입니다. TransCAD에서 스케치를 그리기 위해서는, SketchEditor 객체를 사용하게 됩니다. 그리고 이 스케치 에디터를 사용하기 위해서는 OpenEditor() 명령을 내려줘야 합니다.

여기까지가 실제로 스케치를 그리기 위한 기본 작업입니다. 정리해보면, **1) Reference 객체를 통해 스케치를 할 면을 선택한다. 2) AddNewSketchFeature에, sketch의 이름과, reference를 입력으로 하여, 새로운 스케치 객체를 선언한다. 3) 선언된 스케치 객체를 OpenEditor() 하여 스케치를 작성할 준비를 한다.** 의 순서가 되겠습니다.

다음 6줄은 이제 실제로 스케치를 눈에 보이도록 그리는 작업입니다. Arc, Line, Circle등을 스케치 할 수 있습니다. 위 예제코드에서는 Create2DLine2Points 함수를 이용했네요. 즉, 두 점을 잇는 직선을 그리는 함수를 사용하여, L 모양의 스케치를 그린 것입니다. 그 후, Close() 명령을 이용하여 스케치 에디터를 닫아 주시면, TransCAD 화면에 L 모양의 스케치가 생성된 것을 볼 수 있을 것입니다.

다음은 Extrusion Feature를 생성하는 부분입니다.

짧고 간단합니다. 실은, 번역 과정에서는 이 부분이 가장 복잡하고, 골치 아픈 부분입니다. ^^ 하지만 아직은 예제니까 걱정하지 않으셔도 됩니다.

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);

첫 줄은 bstr\_t name1 = spSketch1->Name; 입니다. spSketch1->Name은 무엇일까요? 아까 spSketch1을 만들기 위해서, AddNewSketchFeature함수를 사용할 때, 스케치의 이름을 입력해 주었죠? 그 이름을 리턴받는 것입니다. 즉 “Sketch1” 스트링 입니다. 이 스트링을 name1이라는 변수에 입력해 주었습니다.

그 후, spBaseSketch라는 reference 형식의 변수를 선언하고, spPart->SelectObjectByName(name1); 을 입력해 주었죠.

이전에 설명드렸듯이, spPart는 TransCAD내의 “새 문서” 안에 있는 Part모델 전체를 관리하는 클래스 입니다. 즉, 지금까지 우리가 생성한 스케치와, reference에 대한 정보들을 이 클래스를 통해서 얻어낼 수 있다고 생각하시면 됩니다. 그러면, SelectObjectByName 함수는 무슨 역할을 할까요? 이름에서 바로 알 수 있듯이, 이름을 입력으로 넣어주면, 그 이름을 가진 Object를 선택해 주는 것입니다. 이름으로 우리는 “Sketch1”을 입력해 주었죠? 그러면 이제 이, spBaseSketch 변수는, “Sketch1”을 나타내는 reference가 된 것입니다.

다음 줄을 보면 이 reference가 어떻게 쓰이는 지 알 수 있습니다. spFeature->AddNewSolidProtrusionExtrudeFeature를 사용했습니다. spFeature는 지금까지 몇 번 나왔었죠? TransCAD 문서 내의 Feature들을 관리하는 클래스 입니다. 이 클래스 내의 AddNewSolidProtrusionExtrudeFeature함수는, Extrusion을 생성해 주는 함수 입니다. 자세한 사항은 역시 참조 문서를 찾아 보면 됩니다. 이 함수에 입력값으로, ("Extrude1", spBaseSketch, 100, Blind, 0, Blind, false) 7개의 파라메터를 입력했네요. 첫번째는 feature의 이름인 “Extrude1”, 두번째는 reference인 spBaseSketch, 세번째는 depth로, 100만큼 extrude하겠다는 것이죠. 나머지 변수에 대한 설명은 참조 문서를 읽어보시면 되겠습니다.

중요한 것은 역시 reference로 무엇을 입력해 주었냐는 것이겠죠? 두번째 입력 변수인 spBaseSketch가 reference가 되겠습니다. 즉, 아까 우리가 그렸던 “Sketch1”을 extrude한 것이지요. 여기까지 코드를 돌려보면 아까 우리가 그려주었던 L 모양의 스케치가 위로 쭉 튀어나오면서 솔리드 모델이 된 것을 볼 수 있습니다.

다음은 두 번째 스케치인 원을 그리는 단계입니다.

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();

원을 그리기 위해선 먼저 어디에 원을 그릴지 선택을 해줘야 합니다. CAD UI에서는 마우스 클릭으로 어느 면에 스케치를 할 것인지 선택하게 되는데요, API를 사용할 때에는 조금 다른 방법을 사용하게 됩니다.

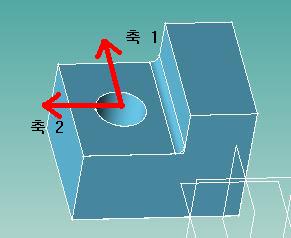
크게 나누어서 Geometry를 이용하는 방법과 Topology를 사용하는 방법이 있습니다. Geometry를 이용하는 방법은, 기하적인 조건으로, 우리가 그릴 평면을 선택하는 방법이지요. 예를들어서, 점 (0,0,0)과, 두 axis벡터인 (1,0,0), (0,1,0)을 입력정보로 주어서 평면을 선택하면, XYPlane이 스케치 평면이 될 것입니다. TransCAD에서도 Geometry 방법을 지원하고 있고, 현재도 송일환씨가 계속 업데이트 중입니다.

두 번째 방법인 Topology 방법이 예시에 나온 방법인데요, 코드에서 SelectBrepByName함수에 들어가는 입력 변수인 ("Extrude1,Sketch1,Line2,0,0,0,ExtrudeFeature:0,0:0;0") 가 핵심적인 부분입니다. 말로 풀어서 보자면 이 변수는 “Extrude1에 의해 생긴 평면중, Sketch1의 Line2에 의해 생긴 평면”을 의미합니다. Line2는 (200,100)과 (100,100)을 연결하는 Line이었죠? 그 Line을 우리가 Extrude1을 통해서 Z축으로 100만큼 Extrude했을 때 생기는 평면이 바로 ("Extrude1,Sketch1,Line2,0,0,0,ExtrudeFeature:0,0:0;0") 라는 이름으로 저장되어 있는 것입니다.

현재 TransCAD에서는 이와 같이, 어떤 형상에 의해 생기는 Face(면)와 Edge(선)에 각각 이름을 붙여 저장합니다. 자세한 이름 붙이기 규칙은 문두환 박사님의 논문을 보셔야 알 수 있습니다만, 번역기 개발 초기 단계에서는 이 예제에 나오는 정도만 이해하여도 크게 무리가 없을 것이라 생각 합니다. Geometry방법을 이용하는 것이 개발자 입장으로서는 간편하겠지만, Geometry 방법은 오류가 발생할 가능성을 기본적으로 내포하고 있기 때문에 각각의 방법이 장단점이 있습니다.

면을 선택한 후에는, 첫 Sketch를 그릴 떄와 마찬가지고 AddNewSketchFeature 함수를 이용하여 IStdSketchFeaturePtr인 spSketch2 객체를 만들어 주었습니다.

다음 단계로, SetCoordinateSystem 함수를 사용하였는데요, 이름 그대로 이 함수는 어떤 스케치의 좌표계를 새로 정의해 주는 함수입니다. 첫 세 인수는 원점으로 사용할 X,Y,Z 좌표, 그리고 네번째에서 여섯번째 인수는 첫번째 축, 일곱번째에서 아홉번째 인수는 두번째 축을 의미합니다. 즉, (150,100,50)을 중심으로 하고, Z축을 첫번째 축, X축을 두번째 축으로 하는 좌표계가 설정되었습니다. 아래 그림과 같이 말이지요.



이후는 앞과 크게 다르지 않습니다. SketchEditor를 열고, 이번에는 Create2DCircleCenterPoint함수는 이용하여, (0,0)위치에 반지름 15 인 원을 스케치합니다. 좌표축을 새로 설정해 주었기 때문에, 위 그림과 같은 위치에 원이 생성됩니다.

이후 과정들은 직접 코드를 실행해 보시고 의미를 파악해 보시기 바랍니다. 기본적인 의미는 모두 설명해 드렸구요, 새로운 함수가 나왔다면, 참조 문서에서 찾아보시면 모든 정보를 얻을 수 있을 것입니다.

이로써 TransCAD 예제 문서를 마칩니다.

1. **Inventor 예시(API)**

다음은 인벤터 API를 이용한 L-Block 예제 코드에 대해 설명을 드리겠습니다. Macro 사용 시스템 담당자는 이 부분을 넘어가도 괜찮습니다.

Inventor API는 TransCAD의 Automation API와 거의 같은 구조를 지니고 있습니다. 따라서, 앞 절의 TransCAD API 예제 문서를 보셧다면, 코드의 흐름도 거의 같다는 것을 알 수 있으실 것입니다. 차이점은, Inventor API는 비주얼 베이직 기반이며, 더 많은 기능을 지니고 있다는 것입니다.

따라서 앞으로 설명드릴 코드들도 비주얼 베이직 언어로 쓰여있으며, 비주얼 베이직 언어를 잘 모르시는 분들은 기본적인 문법을 미리 파악하시는 것이 좋습니다.

인벤터의 API와 다른 CAD 시스템의 API 구조에는 차이가 있을 수 있으나, 기본적인 흐름은 같습니다. 따라서, 한 번 예제 코드를 이해해 보시고, 본인의 CAD 시스템에서 쓰이는 API에 적용시켜서 L-Block을 한번 그려 보시면 많은 도움이 될 것이라 생각합니다.

(전체 코드는 맨 뒤에 있습니다.)

우선 Make\_LBock()이라는 함수를 만들어주었습니다. 제가 원하는 프로그램은 그냥 실행 시키면 바로 L-Block이 화면에 나타나게 하는 프로그램이기 때문에, 따로 Input 변수들이 없습니다. 그리고 Form 구성도 필요가 없지요.

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))

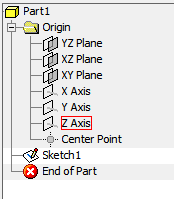
Inventor도 TransCAD와 마찬가지로 Inventor 🡪 Document 🡪 Part Document 🡪 Part( Component Definition) 순으로 컴포넌트를 가집니다. Document 클래스의 함수는 사용할 필요가 많이 없기 때문에 여기서는 Part Document 레벨과 Part Component Definition 단계의 변수만을 지정해 주었습니다.

다음은 oSketch 변수를 선언해 주었는데요, PlanarSketch 형의 변수입니다. TrandCAD의 IStdSketchFeaturePtr와 같은 역할을 합니다. 다음 줄은 조금 자세히 보도록 하지요.

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

우선 oCompDef 변수는 Inventor의 Part 단계를 컨트롤 하는 변수입니다. TransCAD에서의 spPart와 같은 역할을 한다고 보시면 됩니다. oCompDef.Sketches.Add 함수는 현재 생성한 문서 내에서 새 스케치를 추가하는 함수입니다. Add 함수는 1개의 입력 변수와, 1개의 추가적인 입력 변수를 갖는데, 이는 레퍼런스를 찾아보시면 볼 수 있습니다. 첫번째 입력 변수로는 PlanarEntity형의 object를 주어야 합니다. 즉 어느 면에 스케치를 할 것인지를 정하는 변수죠. 위 예제 코드에서 우리는 oCompDef.WorkPlanes.Item(3)를 입력 변수로 주었습니다.

Inventor에서는 기본적으로 3개의 WorkPlane이 주어져 있습니다. WorkPlanes의 첫번째 item, 즉 WorkPlanes.Item(1)은 YZ Plane이고 두번째는 XZ Plane, 세번째는 XY Plane입니다. 아래 그림의 순서와 같죠.



이 외에도 WorkAxis 3개와 WorkPoint 하나가 기본적으로 주어져 있습니다.

Add 함수의 입력 변수로 WorkPlane의 세번쨰 Item을 주었으니 결국 우리는 스케치 평면을 XY 평면으로 잡은 것이 되겠습니다. 이제 스케치를 해야겠죠.

With 문의 쓰임은 Visual Basic 문법에 관한 것이니 설명하지 않겠습니다. ThisApplication은 앞에서도 잠시 나왔었는데요 가장 상위 클래스인 Inventor 자체를 가리키는 클래스 입니다. 그 하위의 TransientGeometry는 기본적인 Geometry들을 생성해 주는 함수들을 포함하고 있는 클래스 입니다. 점, 선, 면, 곡선과 벡터 등을 만들 수 있는데요, 여기서는 점을 만들기 위해서 사용되었습니다. 이렇게 알아두고 우선 다음줄로 넘어가죠.

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

두번째 줄에서는 행렬 변수를 사용해서 SketchLine형 변수를 선언하였습니다. L-Block을 위해서는 6개의 선이 필요하겠죠.

세번쨰 줄부터 8번째 줄까지는 6개의 Line들을 스케치 하기 위한 명령들입니다. 하나씩 보죠. 우선 oSketch는 앞서 선언하였듯이 XY 평면입니다. oSketch(PlanarSketch 클래스)아래에는 많은 개체들이 있는데요, 우리는 Line을 그려야 하므로 우선 SketchLines 클래스로 들어갑니다. 그러면 두 점을 입력 변수로 사용하여 그 두 점을 잇는 Line을 Sketch하는 AddByTwoPoints 함수를 사용할 수 있습니다.(Inventor API Reference 문서를 하나하나 따라가면서 확인해 보세요. Visual Basic에서는 Object Browser를 제공하여 좀 더 쉽게 확인할 수 있습니다.)

하여 oLines 행렬의 1번부터 6번까지를 각각 (0,0),(100,0)을 잇는 oLines(1), oLines(1)의 끝점((100,0))과 (100,50)을 잇는 oLines(2) 등으로 순차적으로 이어 나갑니다.

주의할 점은 만약 두번째 선을 그릴 때에, 위와 같이 하지 않고 다시 (100,0)과 (100,50)의 두 점을 잇도록 그리면 oLines(1)과 oLines(2)가 이어지지 않는다는 것입니다. 구속 조건을 따로 지정해 주어야 하지요. 좀 더 공부하고 싶으시면 구속 조건을 주는 법을 찾아서 해 보시는 것도 좋습니다.

이로써 XY 평면에 L 모양의 스케치가 나타나게 되었습니다.

다음은 이 Sketch를 Extrude 시켜 보겠습니다.

우선 oExtrude\_1이라는 변수를 ExtrudeFeature 형으로 선언해 두었습니다. 이후에 실제 스케치를 Extrude 한 뒤에 그 Extrude의 깊이 등의 파라미터를 바꾸고 싶은 때에는 이 oExtrude\_1 변수로 접근하면 돼겠습니다.

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)

다음은 oProfile이라는 변수를 Profile 형으로 선언해 주었습니다. 이 Profile이라는 것은 TrandCAD에서의 Reference와 거의 비슷합니다. 좀 더 쉽게 이것이 무언인지 알아보려면 이전 단계까지만 코드를 실행한 뒤에 Inventor의 인터페이스에서 Extrude 버튼을 눌러보시기 바랍니다. Profile은 두 가지로 크게 나뉘는데요, 하나는 예제에 있는 것처럼 AddForSolid, 즉 스케치의 내부를 Extrude 하는 것이고 하나는 AddForSurface, 가장자리 선을 Extrude 하는 것입니다. 지금은 AddForSoild를 사용해야 되기 떄문에 스케치 변수인 oSketch의 Profile에 접근하여 AddForSolid 함수를 호출해 주고, 이를 oProfile에 저장해 줍니다.

다음은 아까 선언했던 oExtrude\_1에 실제 Extrude를 저장해 주는 코드입니다. TrandCAD와 비슷하게 oCompDef의 Feature 클래스 내의 ExtrudeFeature로 접근하여 AddByDistanceExtent 함수를 호출해 주면 됩니다. 입력 변수는 레퍼런스 문서를 참조하여 입력해 주면 됩니다. 여기서는 아까 저장했던 Profile과 Extrude할 거리, 방향, Joint Operation을 선택하였습니다.

추가적으로 언급하고 싶은 것은, 위 코드에서 oCompDef.Features.ExtrudeFeatures.AddByDistanceExtent(oProfile, 80, kPositiveExtentDirection, kJoinOperation) 부분만 적어도 Inventor 내의 문서에 Extrude가 적용됩니다. 이것을 oExtrude\_1이라는 변수에 저장하는 것은 후에 Extrude로 인해 생성된 Face나 Edge에 접근을 좀 더 쉽게 하기 위한 것이라는 것을 알아두셧으면 합니다.

다음은 마지막인 Circle 스케치 부분입니다.

TransCAD 부분과 마찬가지로, 스케치할 원의 중심점을 원점으로 하는 Coordinate를 먼저 설정해 주겠습니다. Inventor에서는 이전에 설명드린 바와 같이 기본 WorkPoint를 제공하고, 이 점은 (0,0,0)입니다. 다른 점을 원점으로 하는 Coordinate를 설정해주려면 우선 WorkPoint를 하나 추가해 주어야 합니다. 우선은 TransientGeometry 클래스 내의 CreatePoint 함수를 호출하여 (70,50,40)에 점을 하나 만들어 준 뒤, oWorkPoint 변수를 생성하여 이 점을 WorkPoint로 추가해 줍니다.(oCompDef.WorkPoints.AddFixed(oPoint))

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

그런 뒤에 새 스케치를 추가해 주는데요, 마찬가지로 oCompDef.Sketches.Add 함수를 사용하는데요, 여기서는 입력 변수로 oExtrude\_1.SideFaces.Item(4)를 사용하였습니다. 아까 oExtrude\_1 부분을 설명하면서 이 변수를 지정한 것은 Face로의 접근을 쉽게 하기 위해서라고 말씀 드렸었죠? 보시는대로 우리가 원을 스케치할 평면을 선택할 떄에 oExtrude\_1의 SideFace들 중 4번째 평면을 선택하면 됩니다.

그 뒤에 oSketch의 원점이 저장되어있는 OriginPoint로 접근하여 우리가 아까 설정한 (70,50,40)의 WorkPoint를 원점으로 바꾸어 줍니다. 그런 뒤에는 oSketch 내의 SketchCircles클래스의 AddByCenterRadius 함수를 호출하여 입력 변수를 원점과 반지름을 주어 원을 그려주게 됩니다.

인벤터의 API에 관한 설명은 여기까지 입니다. 기본적으로 Inventor는 TransCAD API와 비슷한 구조를 가지고 있기 때문에 특별히 다른 점이 많지 않습니다. 그러나 다른 CAD시스템의 API를 담당하시는 분들은 넓은 관점에서 볼 때의 기본적인 API구조는 비슷하지만 상세히 보면 많은 다른 점이 있을 것입니다.

따라서 제가 설명한 부분은 한 번 읽어보시고, 실제 본인의 CAD 시스템 내에서 API로 Part를 생성해 내는 것은 추가적인 공부가 필요합니다. API를 제공한다면 기본적으로 매뉴얼과 함수들의 Reference 목록은 제공하기 때문에 열심히 공부하신다면 크게 어려운 일은 아닐 것입니다.

1. **SolidWorks 예시(Macro)**

SolidWorks는 Macro를 사용하는 CAD 시스템 중의 하나입니다. Macro는 보통 Text 형식으로 저장되며, CAD 시스템 내에 이 Text파일을 저장하거나 불러오는 기능이 포함되어 있습니다. Macro 파일을 저장하기 위해서는 특정 명령을 통해서 Macro 파일 저장 모드를 시작하고, Part를 손으로 그리게 되면 그에 해당하는 Macro 파일이 생성됩니다.(CAD 시스템마다 다를 수 있습니다.) 이 Macro 파일을 CAD 시스템 내에서 다시 불러와서 실행시키면, 손으로 모델링 한 순서 그대로 자동으로 CAD 시스템이 Part를 그려냅니다.

앞의 예제(Inventor API)와 마찬가지로 각각의 CAD 시스템에 따라서 세부적인 명령어와 동작 방식이 다를 수 있습니다. 본 예제는 하나의 예시로만 참고하시기 바라며, 각자의 시스템이 어떻게 이루어져 있는지는 본인이 익히셔야 합니다. 하지만 유사한 구조 또한 존재하므로, 이 가이드를 한번 읽어 보시는 것을 권장합니다.

이 Chapter에서는 SolidWorks의 L-block 예제 Macro 파일을 가지고 Macro 전반에 대해 설명을 하도록 하겠습니다.(전체 코드는 맨 뒤에 있습니다.)

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

처음 부분은 변수 선언으로 시작압니다. Dim과 Set 등의 명령어를 보면 SolidWorks는 VB(Visual Basic) 언어를 기반으로 하는 매크로 파일을 사용하는 것을 알 수 있습니다. 여기서 중요한 변수는 swApp, Part, SelMgr, Feature 입니다. 각각 이름으로 유추해 보면, swApp는 SolidWorks 캐스 시스템 자체를 컨트롤하는 변수, Part는 부품 단위, Feature는 Feature의 변수이고, SelMgr은 아마 Select(즉, Reference의 선택)를 관리해 주는 변수인 듯 합니다.

다음은 Sub main() 이네요. Visual Basic의 메인 함수입니다. 기본적으로 Visual Basic 언어이기 때문에 Visual Basic에 대한 이해가 조금 있으시면 이해하시기가 더 편할 것 입니다.(CATIA Macro 또한 Visual Basic 기반입니다.)

다음은 swApp라는 변수를 Application의 SldWorks로 Set 해 주었습니다. 실제 내 컴퓨터에서 실행되는 SolidWorks CAD 프로그램을 컨트롤 할 수 있도록 swApp 변수를 선언을 해 준 것입니다. Part 변수는 swApp의 ActiveDoc 형식으로 선언해 주었습니다. 현재 Active(활성화) 상태인 document를 말하는 것이겠죠? 그리고 SelMgr은 Part의 SelectionManager 형식으로 선언해 주었습니다.

사실 이 부분은 앞서 TransCAD, Inventor 예제와 마찬가지로 개발자들이 크게 신경쓸 부분은 아니라고 할 수 있겠습니다. Part 수준의 변환에 있어서 이 부분은 어떤 Part를 디자인하던지 변화가 없는 부분이기 때문입니다. 특히 Macro를 사용하는 시스템 같은 경우, 사용자가 이제부터 매크로 입력을 하겠다는 신호를 프로그램(CAD 프로그램)에 내려준 뒤, 손으로 Part 설계를 하면 자동으로 매크로 파일이 생성되는 식의 시스템에서는 더더욱 그렇습니다.

다음 부분부터가 본격적으로 L-Block을 생성하는 부분입니다.

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

Boolstatus 변수는 앞서 매크로의 시작부분에 선언된 변수입니다. 일반적으로 내가 내린 명령이 성공하면 true, 실패하면 false가 반환되는 함수의 출력값을 담기 위해 사용되는 것입니다.

호출한 함수를 살펴보면 SelectByID2 멤버 함수인데 이는 Part 클래스 하위의 Extension 클래스의 멤버 함수입니다. 이름을 보면 알겠지만, 어떤 geometry를 ID를 통해서 선택하는 기능을 하는 함수입니다. TransCAD, Inventor에도 마찬가지 기능을 하는 함수가 있었죠? 즉 처음에 스케치를 시작할 XY 평면을 선택하기 위해서 함수를 호출한 것을 알 수 있습니다. 입력 인자로 “정면”과 “PLANE” 등등이 입력되었네요. 자세한 것들을 알고다면 SolidWorks 자체의 API 문서를 살펴보아야 합니다. 우선은 매크로 파일의 해석을 위한 설명이므로, 정면의 Plane을 선택한 것이라고 보면 되겠습니다.

함수 이름 뒤에 2가 붙은 것은, 버전간 매크로 파일의 호환성을 유지하기 위해서 입니다. 즉 예전 버전에서 SelectByID라는 함수를 사용한 매크로 파일이 있었는데, 버전이 업그레이드 되면서 이 함수를 개선하였을 때, 같은 SelectByID라는 함수명을 사용한다면, 개선되면서 내부의 동작 방식이 바뀌었을 수도 있고, 입력 인자들의 변화가 생겼을 경우에는 아예 이전 버전의 매크로 파일이 제대로 동작하지 않을 수가 있습니다.

따라서 새 버전에서 함수를 개선하고자 할 때에는 SelectByID2라는 이름을 사용하여, 나중에 SelectByID3, SelectByID4 라는 개선된 함수들이 추가되어도 정상적으로 작동할 수 있도록 하기 위함입니다.

다음은 InsertSketch2라는 함수를 True 상태로 만들어 주었네요. 즉 스케치를 시작하겠다고 선언해 준 것입니다. 다음줄들은 이제 별 설명이 없어도 아시리라 생각압니다. Line 스케치들을 마친 뒤에는 ClearSelection2을 해 준뒤 InsetSketch2 함수를 호출하였습니다.

이로써 L 모양 스케치는 끝났고 그 다음 부분을 봅시다.

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

이번에도 SelectByID2 함수를 사용해 무언가를 선택해 주었네요. 입력 인자를 보면 “선1@스케치1”과 “EXTSKETCHSENGMENT”입니다. 즉 방금 한 스케치의 첫번째 Segment Line을 선택해 준 것이지요. 그리고 그 뒤의 0.005는 아마 x좌표의 값인것 같습니다.

선택이 되었으니 Extrusion을 해 주어야겠죠? Extrude는 Part 클래스 하위의 FeatureManager 클래스 밑의 FeatureExtrusion2 함수를 호출하여 Feature를 만들어 주었네요. 그리고 EnableContourSelection을 false( = 0) 으로 만들어 주었습니다. 무슨 역할을 하는지는 잘 모르겠네요. 일단 넘어가겠습니다.

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

다음은 원 스케치 부분입니다. 그런데 SolidWorks에서는 Extrude한 결과로 생긴 Face에 이름을 붙이지 않는 모양이네요. 이름으로 선택하지 않고, (0.0065,0.006,0.004)라는 점을 입력으로 선택 받았습니다. “FACE”라는 것은 명시를 해 주었구요. 그 뒤에는 앞서 L 모양 스케치와 마찬가지로 InsertSketch2와 ClearSelection2 함수, 그리고 원을 스케치 하기 위한 CreateCircleByRadius2 함수를 사용해서 원을 스케치 해 주었습니다.

참고로 SolidWorks에서는 지역 좌표계(local coordinate)를 지원하지 않기 때문에, 원을 그릴 때에 절대 좌표인 (-0.0065,0.004,0)을 중심으로 입력하였습니다. Inventor에서는 중점의 위치를 잡아준 뒤, local coordinate를 이용하여 (0,0)에 원의 중점을 잡아주었지요.

이러한 시스템의 차이를 파악하는 것이, CAD에 대한 이해뿐만 아니라, 번역기를 개발하는 데에 있어서 중요합니다. 번역기는 서로 다른 시스템 간의 정보 교환을 목적으로 개발되는 것이기 때문에 당연하겠죠?

다음은 마지막 부분입니다. SolidWorks 매크로는 매우 간결하게 구성되어 있어 길이가 상대적으로 짧습니다.

역시 SelectByID2 함수를 사용한 선택으로 시작합니다. 원을 Cut Extrude 하기 위한 과정이기 때문에 아까 했던 Sketch를 선택해야 합니다. 원은 Arc의 일종으로 간주하는 모양입니다. “Arc1”을 이름으로 하는 “SKETCHSEGMENT”를 선택하였습니다. 여기서는 “EXTSKETCHSEGMENT”가 아닌 그냥 “SKETCHSEGMENT”를 선택하였습니다. 차이점이 있을 것이라고 생각되지만 CAD 시스템 자체에서 정의되는 것이므로 깊게 파고 들어가지는 않겠습니다. 우선 이 매크로 파일은 잘 만들어진 파일이기 때문에 문제점이 없습니다.

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

다음은 이전의 Extude처럼 FeatureManager클래스 하위의 FeatureCut 함수를 호출하여 Cut Extrude를 Part에 생성합니다.

마지막 남은 Fillet Feature는 역시 SelectByID2 함수를 사용해서 이름이 없는 “EDGE”를 좌표값 (0.003027566077776, 0.005978589108679, 0.005298091073996)를 통해서 선택해 주고, FeatureFillet5 함수를 호출하여 생성해 줍니다.

매크로 파일은, 보면 알 수 있듯이 사용자가 설계 하는 과정을 그대로 담고 있습니다.(모든 매크로 시스템이 다 같은 것은 아니지만) SolidWorks같은 경우엔 앞의 Inventor, TransCAD와는 다르게, 선택을 한 이후에 선택한 면의 정보를 Reference 변수에 저장한 뒤에 Feature로 넘기지 않습니다. 따라서 번역을 하는 과정에서 이런 부분을 어떻게 처리해 주어야 할 지에 대해서도 생각을 해 보아야 할 것입니다.

1. **변환 방법**

앞의 예제들은 통상적으로 API나 매크로 파일을 통해 각각의 시스템에 Part를 “그리는” 법에 대해서 알아 본 것입니다. 변환기를 구현하기 위해서는 “그리기” 이전에 “읽기” 능력이 필요합니다. 즉, A 시스템에서 B 시스템으로 변환을 한다고 할 때에, B 시스템에 그리기 이전에, A 시스템에 그려져 있는 것을 읽을 수 있어야 합니다.

이 장에서는 간단히 두 가지의 예제 코드를 통해서 변환기가 전체적으로 어떻게 흘러가야 하는지, 그리고 주요하게 해결해야 할 문제점이 무엇인지 알아볼 것입니다.

변환기에 대한 예제 코드는 첨부하지 않으며, 본인이 맡게 된 시스템을 먼저 개발한 분에게 코드를 받아서 개별적으로 분석해 보셔야 합니다.

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

이 코드는 Inventor의 Part를 TransCAD의 Part로 변환하기 위한 기본 틀을 보여주는 코드입니다.

Macro-parametric 방법은 설계 History를 보존해야 하기 때문에 기존에 설계된 순서대로 번역을 해야 합니다. 앞서 L-Block 도면을 그리는 순서를 생각해 보면

XY Plane 선택 🡪 L 모양 스케치 🡪 Extrude Feature 생성

면 선택 🡪 원 스케치 🡪 Cut Extrude Feature 생성

Edge 선택 🡪 Fillet Feature 생성

의 세 단계를 통해서 part가 디자인 되었습니다. 따라서 이를 변환하기 위해서는 L-Block을 그리기 위해 사용된 세 가지 Feature(Extrude, Cut Extrude, Fillet)를 TransCAD에서 재연해 주어야 합니다.

이제 옆의 코드를 보면, oFeature를 입력으로 받아서 Feature의 Type을 체크한 후, 각각의 Type에 따라서 변환하는 함수를 호출해 주는 것을 볼 수 있습니다. 따라서 L-Block에서는 ExtrudeFeature, ExtrudeFeature, FilletFeature 순서로 Feature가 들어올 것입니다.

그러면 이 순서대로 그냥 TransCAD에서 그려주는 함수를 호출해 주면 변환기의 구현이 끝날 것 같아 보입니다. 이론적으로는 맞습니다만 조금 어려운 점이 있습니다.

우선 위의 첫 단계인

XY Plane 선택 🡪 L 모양 스케치 🡪 Extrude Feature 생성

부분을 번역하는 과정을 생각해 봅시다.

Extrude Feature 체크(From Inventor) 🡪 **Extrude Feature에 사용된 Sketch 체크 🡪 Sketch의 Reference 체크 🡪 Sketch의 각각의 Entity(점, Line, Arc 등) 체크 🡪 Sketch의 Reference 생성(in TransCAD) 🡪 Sketch의 각각의 Entity 생성 🡪 Extrude Feature에 사용될 Sketch 선택 🡪 Extrude 생성**

의 과정을 거쳐야 합니다. 따라서 옆 예제 코드의 INV\_PRE\_ExtrudeFeature 함수는 Extrude Feature 체크 부분을 제외한 나머지 과정을 모두 수행해 주어야 합니다. Sketch의 Reference 체크, Feature에 사용된 Sketch 체크 등은 모든 Feature마다 거의 필요한 부분이므로 별도의 함수로 구현하여 호출하는 것이 좋습니다.

그러나 각각의 Feature, Sketch마다 모두 다른 특성을 가지고 있기 때문에 이러한 부분을 일반적으로(즉 L-Block 이외의 다른 모델들도 번역 가능하도록) 구현하는 것은 쉽지 않습니다. 하지만 최대한 일반적으로 번역을 할 수 있도록 구현을 하도록 노력해야 합니다. 그렇지 않다면 모델들마다 새로운 번역기를 개발해야 한다는 것인데, 이는 아무런 의미가 없습니다.

TransCAD를 Inventor로 번역하는 틀도 크게 다르지 않습니다.

역시 Feature 단위로 어떠한 Feature들이 있는 체크를 하고, 순차적으로 함수를 호출하여 Inventor에 스케치와 Feature를 생성하는 것입니다.

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

2010년, 송일환 선배님이 구현하고 계시는 Hybrid 방식의 Naming 방법과 일반적인 CAD 시스템에서 사용되고 있는 두 가지 Naming 방법인 Geometry-based, Topology-based 방법은, 번역 과정 중, Reference 검출에 있어서 아주 중요한 이슈입니다.

L-Block을 번역하는 과정에서, 원을 스케치 할 때에 스케치를 할 면을 선택하는 과정은, 사용자의 입력(마우스 클릭)을 사용한다면 매우 간단한 문제이지만 자동적으로 변환을 하기 위해서는 여러가지 복잡한 문제점들이 존재합니다. 이는 여러분이 번역기를 개발하다 보면 공감할 수 있을 것입니다.

어떠한 방법을 사용하여 구현을 할 지는 단순히 개발자의 능력에만 의존적인 것은 아니고, Macro-parametric 프로젝트가 어떤 방향으로 진행되어 나아갈 것인지에 따라서도 달라질 수 있습니다.

1. **UG Example (Macro)**

본 장에서는 UG의 Macro 파일을 이용한 번역에 대해 설명하겠습니다. Macro파일은 설계자가 이용한 모델링 명령어의 이력을 담고 있습니다. Macro파일 도는 로그 파일은 절차적 모델을 반영합니다. 대부분의 사용 CAD시스템이 Macro 파일 포멧을 지원합니다. Macro 포멧은 CAD시스템에 따라 상이하며 UG의 Macro 파일은 GUI 사용자 그래픽 인터페이스 단에서의 Text파일들로 이루어져 있습니다.

먼저 UG에서 Macro옵션으로 전환을 하고 그 상태에서 L-Blcok 을 그려보았습니다. 저장된 Macro파일은 아래와 같은 모습의 Text 파일입니다.

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

Macro파일을 자세히 살펴봄으로써, 우리는 Macro 파일이 모델링 명령어 뿐만 아니라 포인팅 디바이스로 선택된 엔터티의 화면 좌표까지 기록하고 있다는 것을 알수 있습니다. 또한 Macro파일이 모델 수정 이력과 기본 설정값 뿐만 아니라 Undo/Redo 이력까지 기록하고 있다는 것을 알수 있습니다. 이러한 Macro파일을 번역하는 것은 매우 어렵습니다. 그렇기에 UG 번역기도 API를 이용한 번역을 구현하고 있습니다.

1. **UG Translator(API) 셋업 및 사용방법**

본 장에서는 앞선장과는 달리 코드에 대한 내용이 아닌 현 시점에서의 번역기를 어떻게 초기 설치하고 사용하는지에 대한 설명을 담습니다.

각각의 번역기는 특색이 다 다르지만 UG의 경우 구동환경의 영향을 많이 받기 때문에 환경을 일치시키기 위해 Vmware라는 가상머신 툴을 이용합니다. 가상머신을 이용하면 사용자의 PC가 32비트 환경이든지 64비트 환경이든지 혹은 Win XP인지 Win7인지에 상관없이 동일한 환경의 OS에서 구동이 가능합니다. 또한 CAD버전의 다름에 따른 호환성 문제도 해결할 수 있습니다. 다만 너무 오랜시간 업데이트를 하지않았다면 업데이트를 한번씩은 해야할 것 입니다.

UG 가상머신의 경우 CAD시스템으로는 UG NX 7.5버전을 사용하고 있고, OS로는 Windows7 32bits를 이용하고 있습니다.

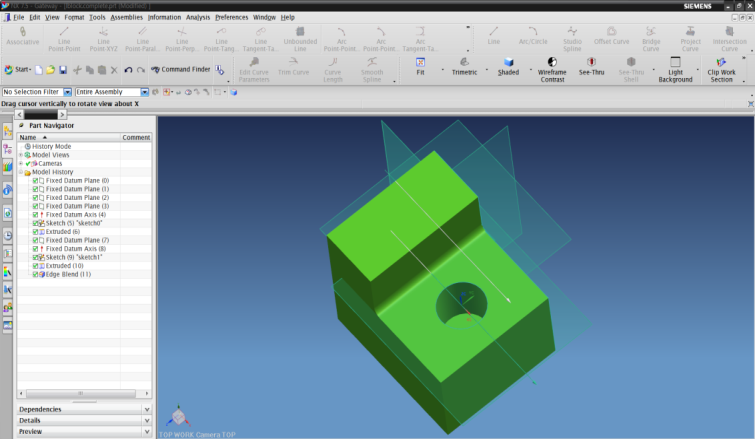


Figure UG NX 7.5 구동화면

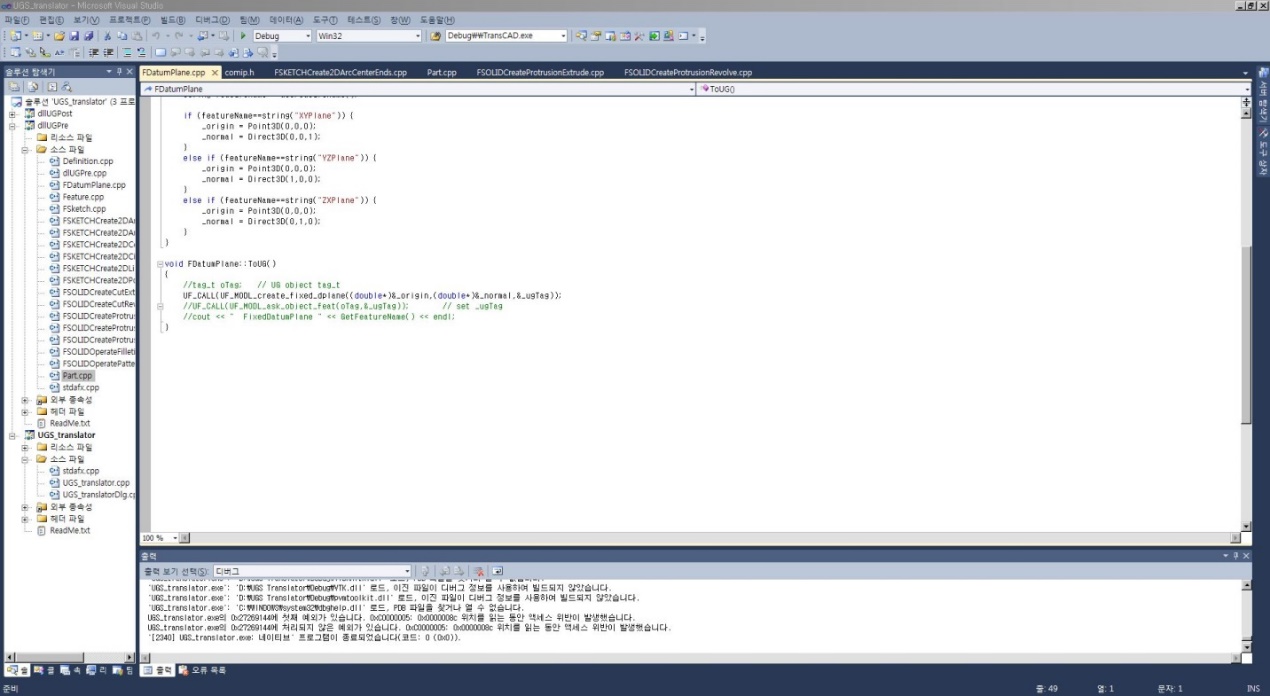
무료로 공개되어있는 Vmware player를 받아서 해당 가상머신을 구동하면 됩니다. OS+번역기+CAD+개발툴(Visual Studio)등이 포함되어있습니다.

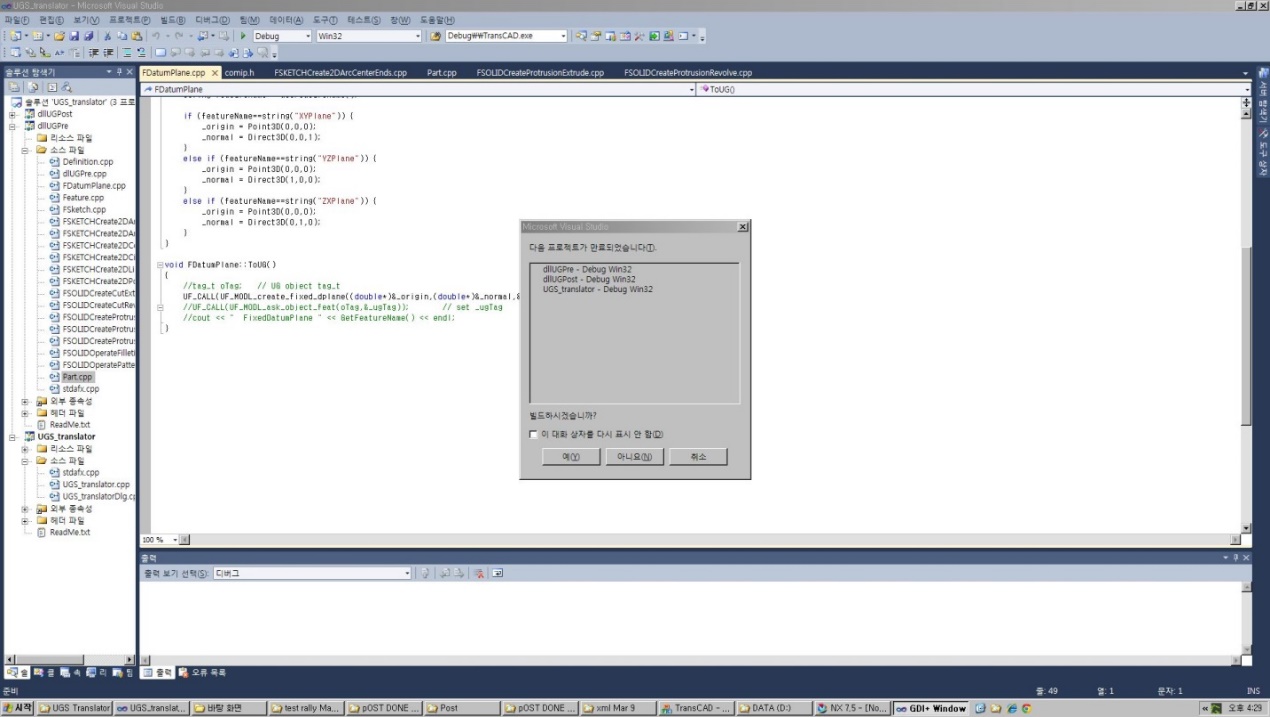
사용자는 확장자가 vmx로 되어있는 파일을 VMware Player상에서 “Open a Virtual Machine” 명령을 통해서 실행하면 됩니다.

가상 머신 내부에 있는 UG번역기를 실행시키는 순서는 다음과 같습니다.

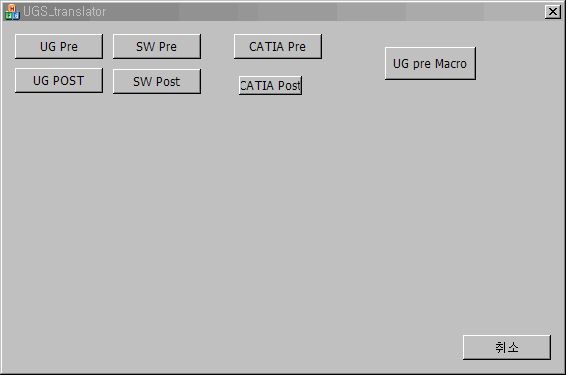
<UG Pre>

1. Click run/debug

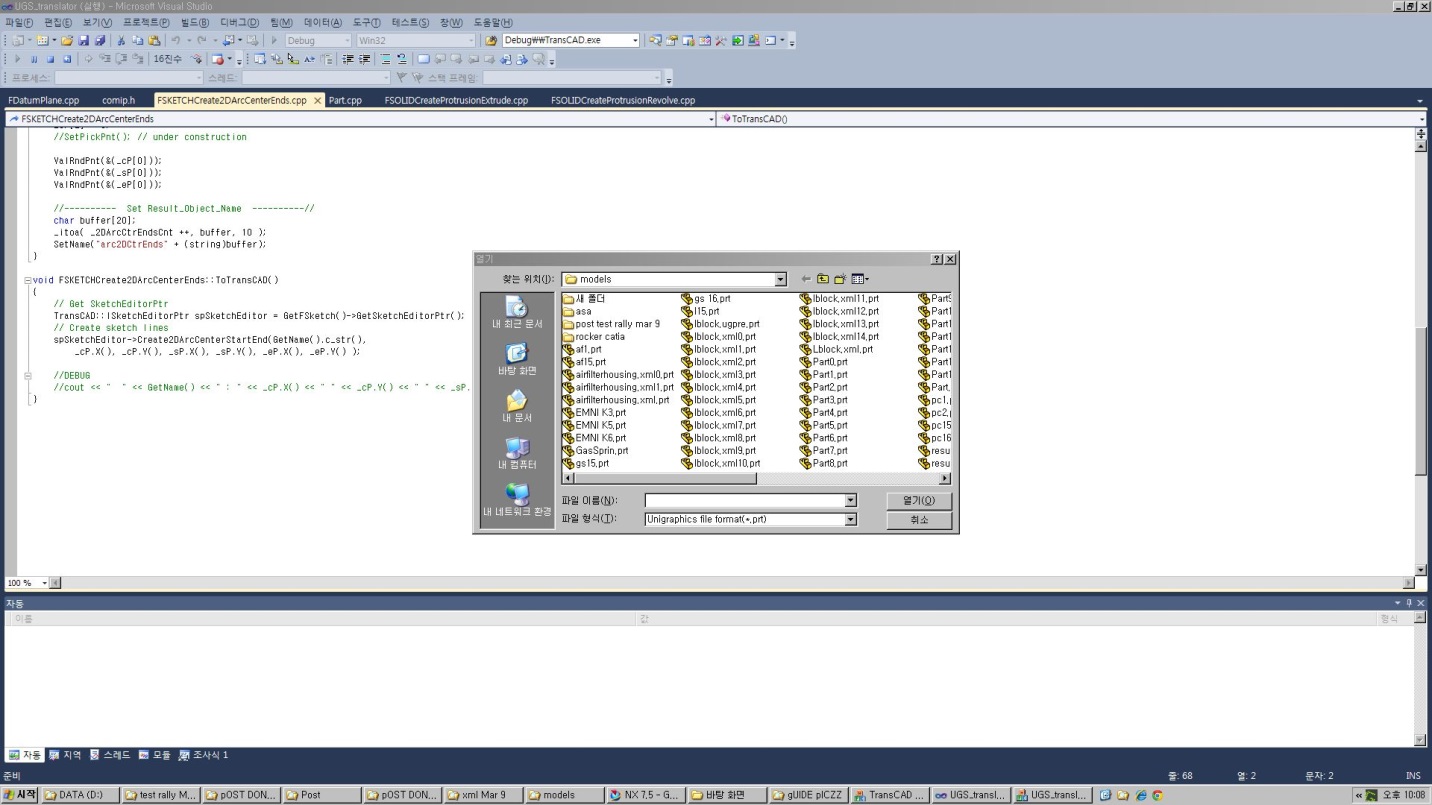
****



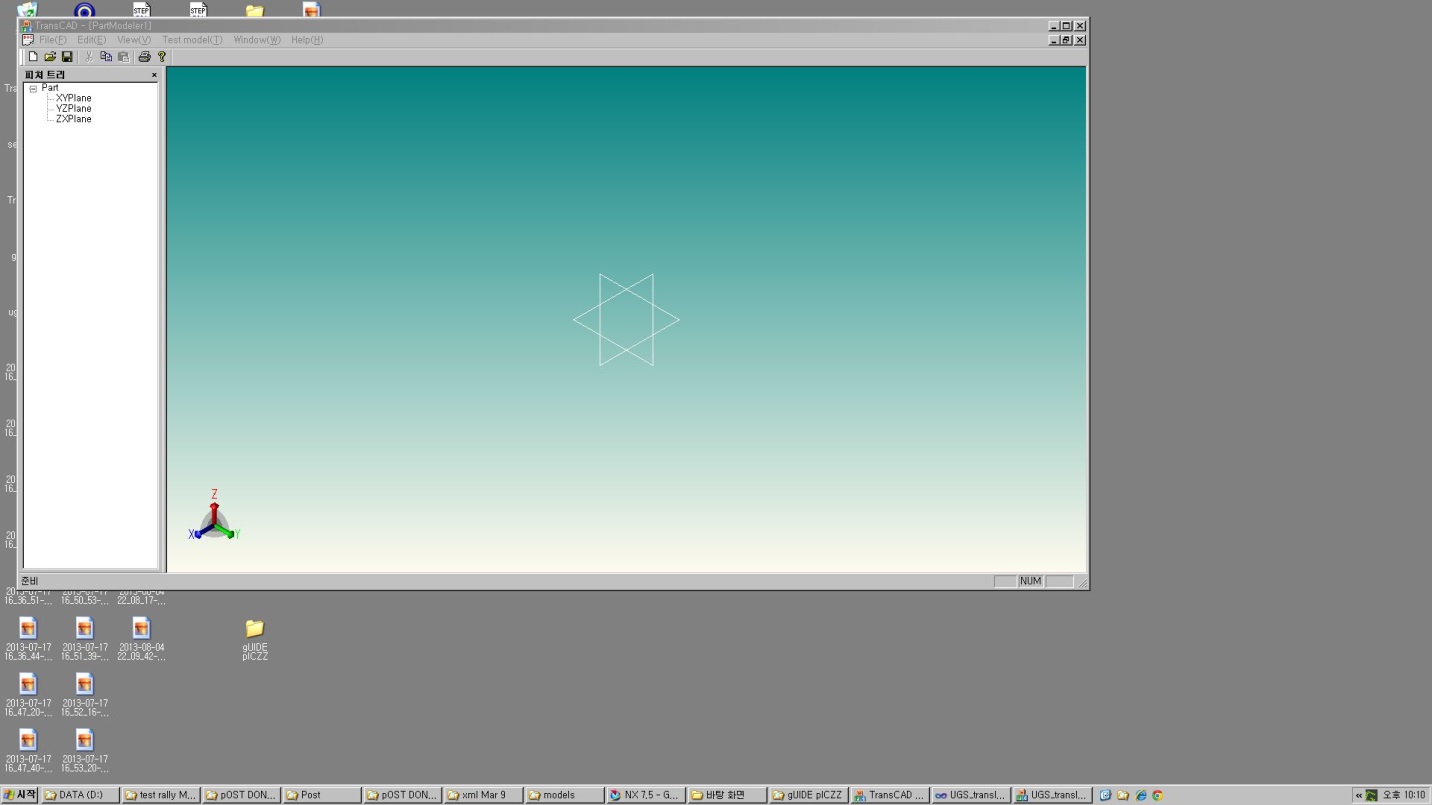
2. Click run/UG Pre



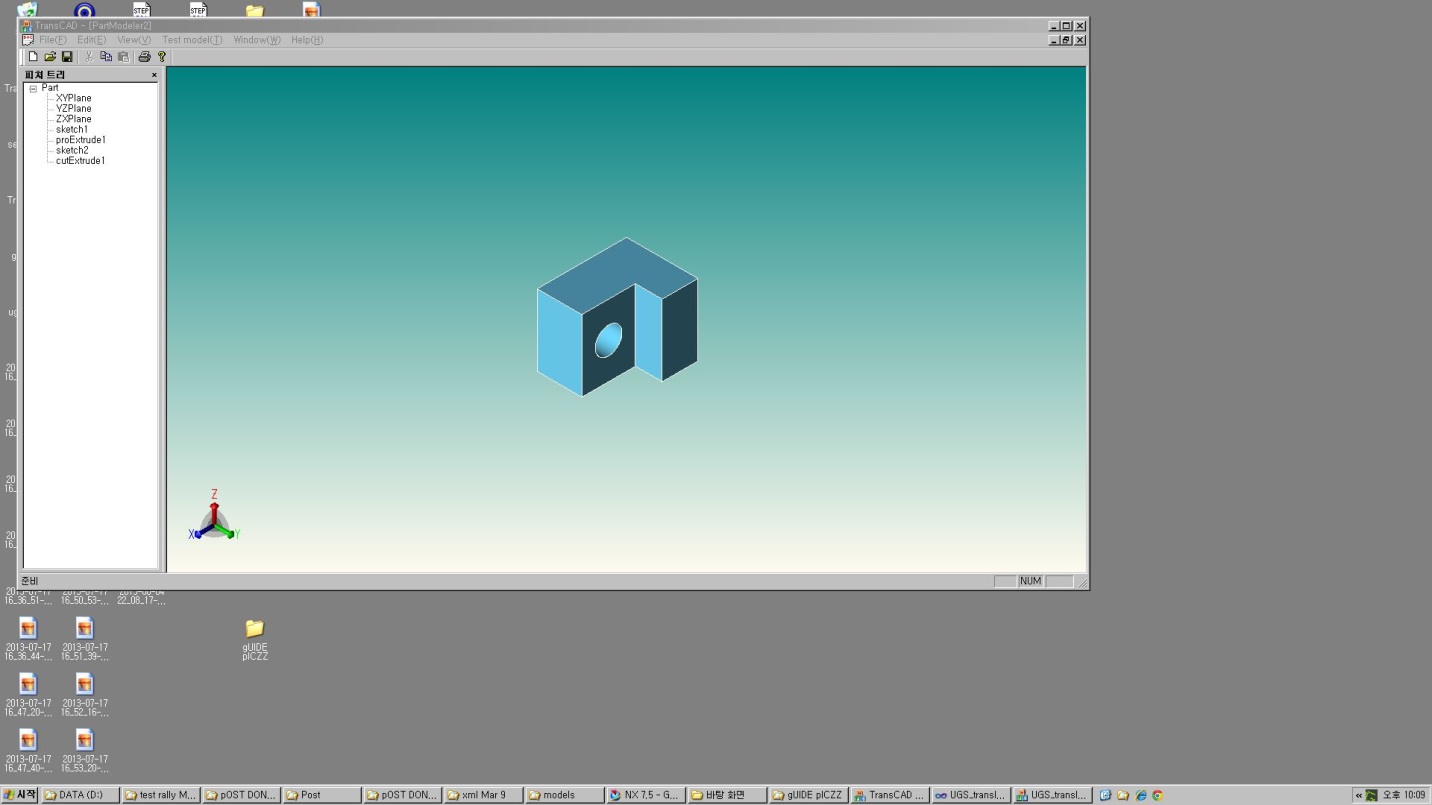
3. Open UG part file



Open ”LBlockUGPre.prt” file

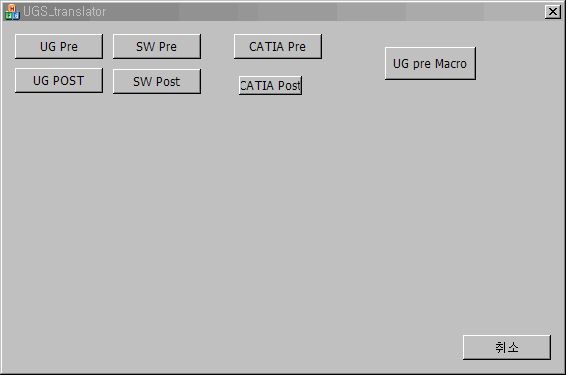


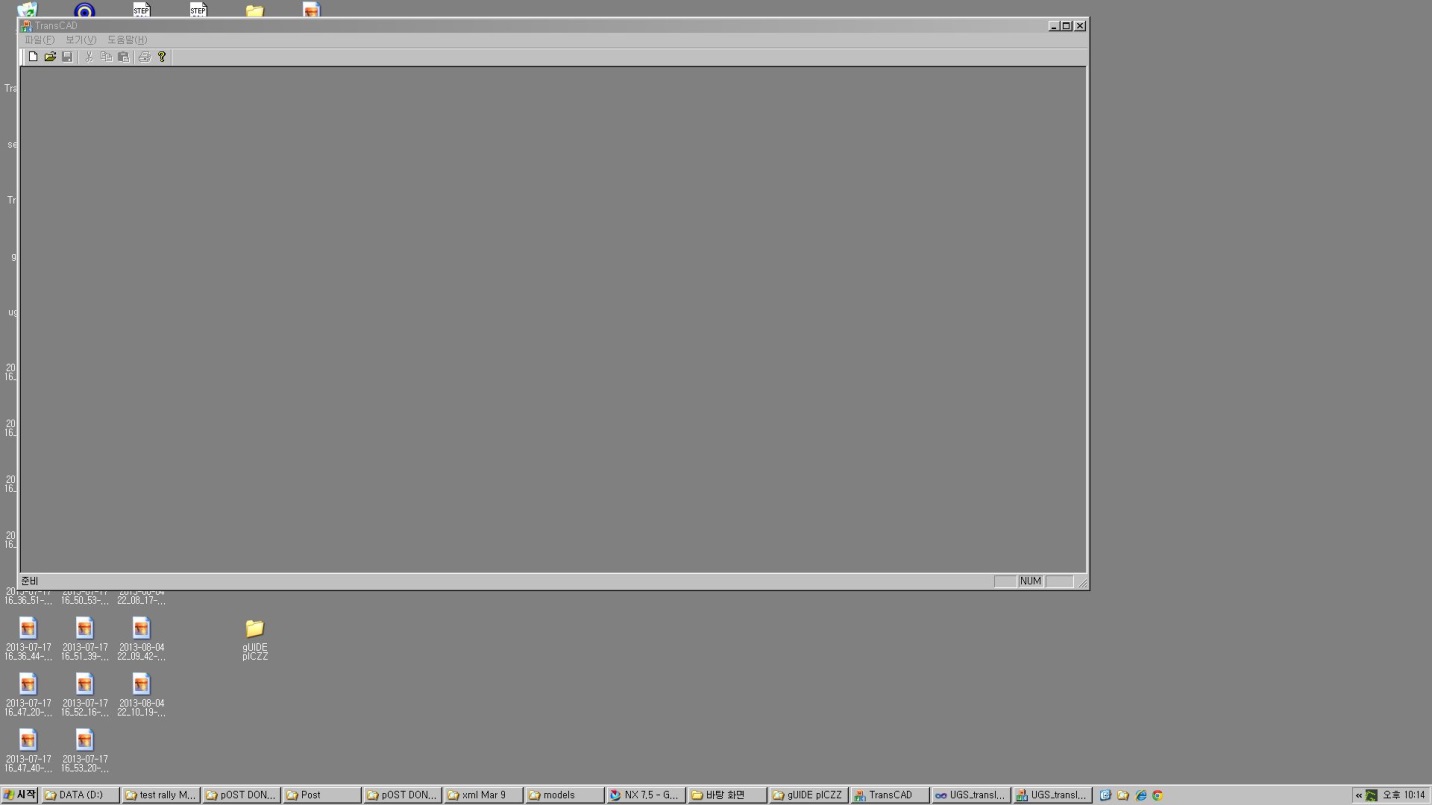
Before in TransCAD



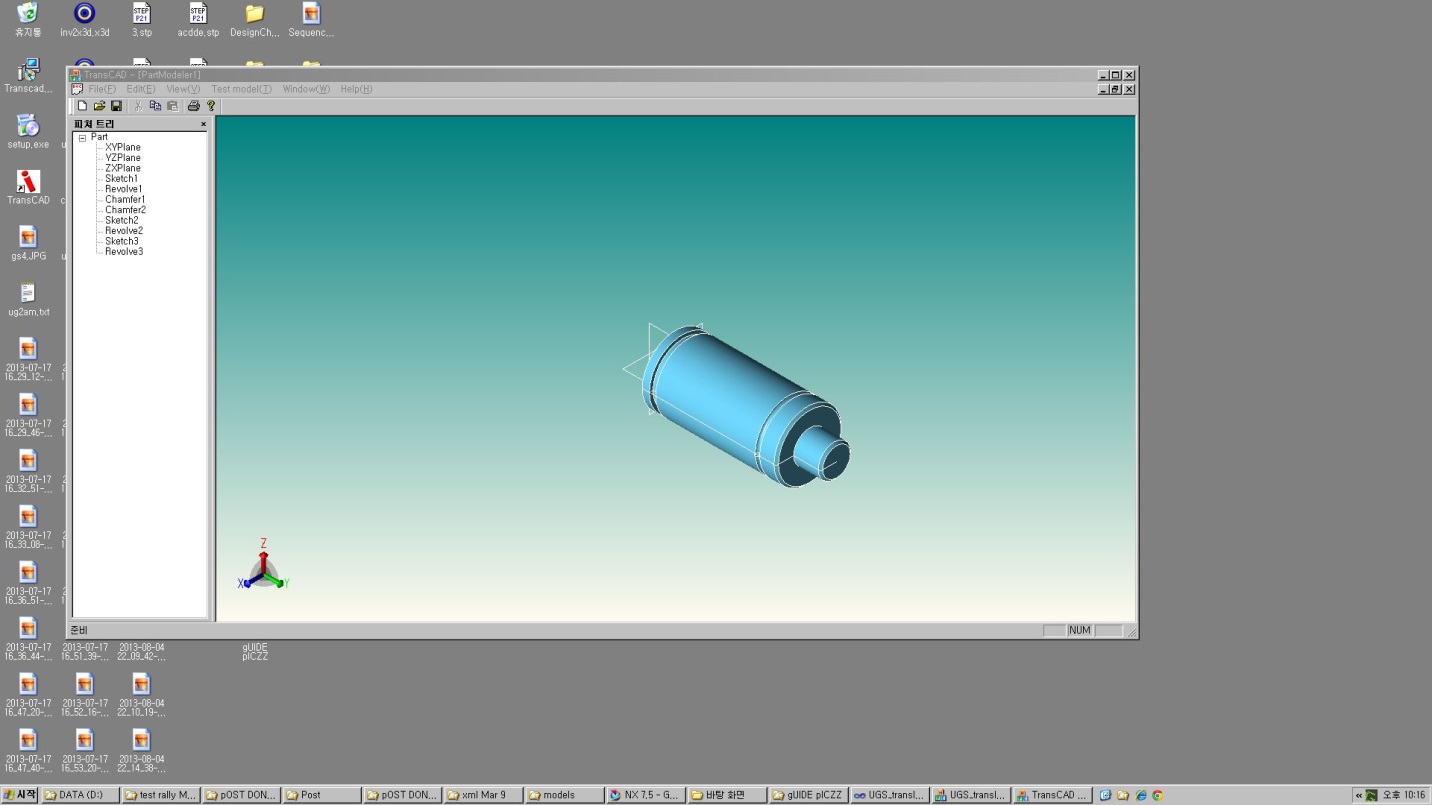
After in TransCAD

<UG Post>

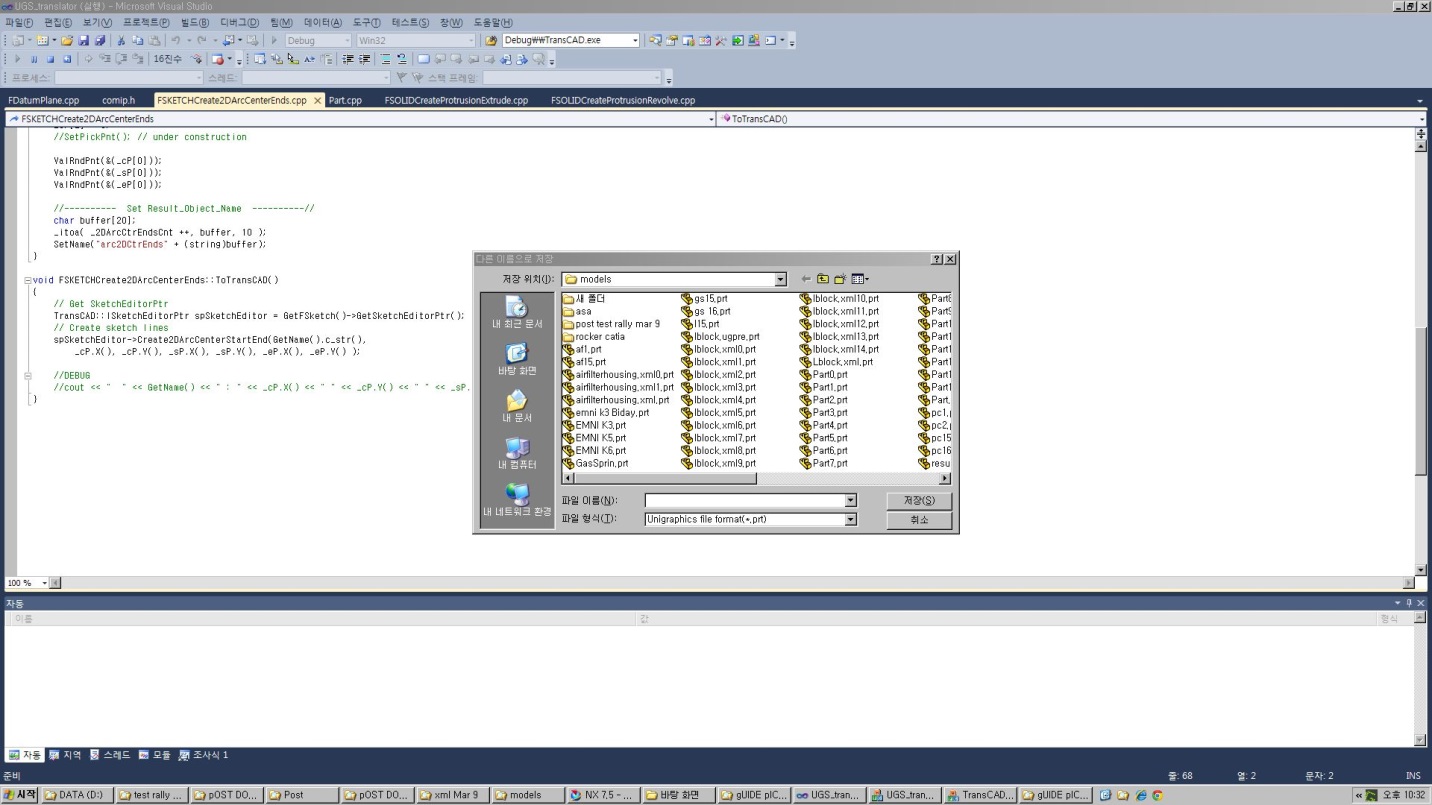




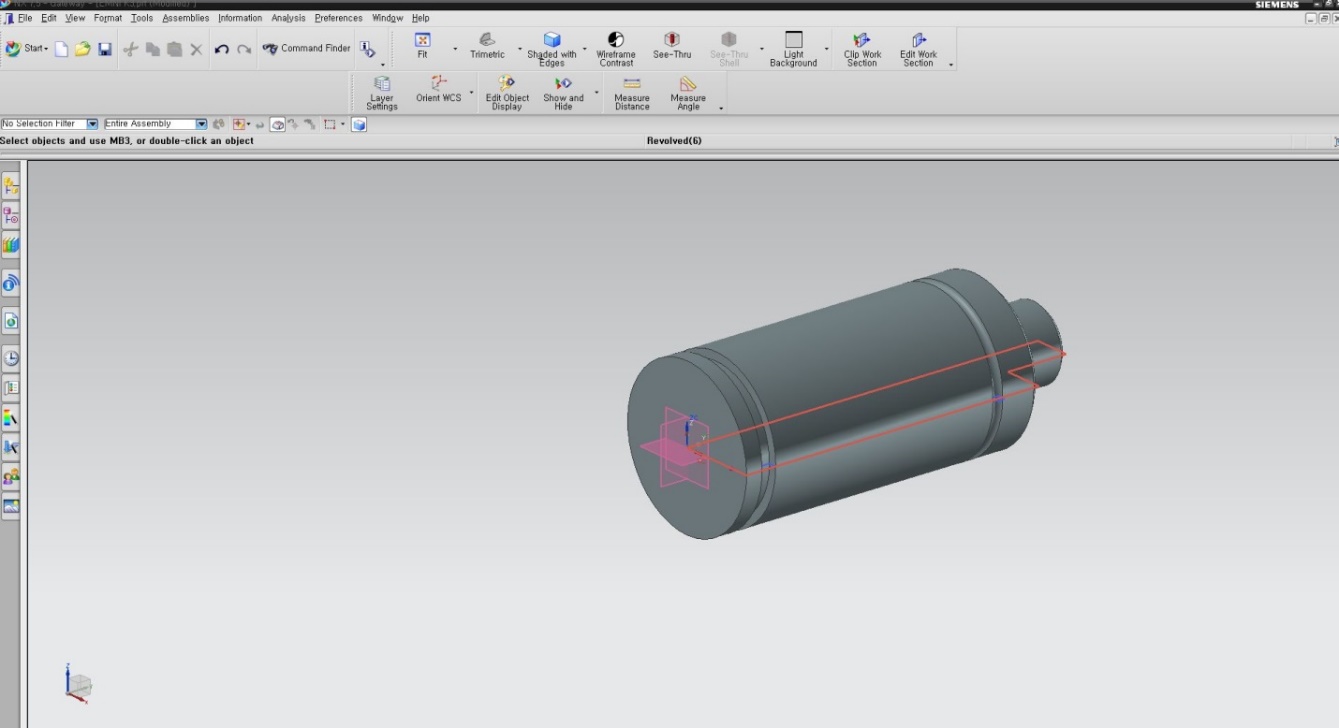
**TransCAD**



**Open Gas Spring in TransCAD**



**Save As GasSpringUGPost.prt**



**Open in UG NX**

1. **맺음**

이로써 Implementation Guide의 작성을 끝마치겠습니다. 자신이 담당하고 있는 시스템에 대하여 가장 많은 도움을 줄 수 있는 것은 이전 개발자라는 것을 알아야 하며, 그들의 코드에 대한 분석이 선행되어야 한다는 것을 잊지 마시기 바랍니다.

또한 매크로를 사용하는 시스템이던, API를 사용하는 시스템이던, 자신이 담당하는 시스템에 대한 이해가 선행되지 않으면 변환기의 구현은 절대 불가능 합니다. 따라서 이 Guide뿐만 아니라, 각자의 시스템에 대한 Guide를 꼭 찾아서 읽어보시기를 권장합니다.

열심히 하세요 ㅋ