초보자를 위한
ArchiCAD 애드온 개발 가이드

ArchiCAD 19 버전 기준입니다.

Add-on 개발에 대한 학습 자료입니다.

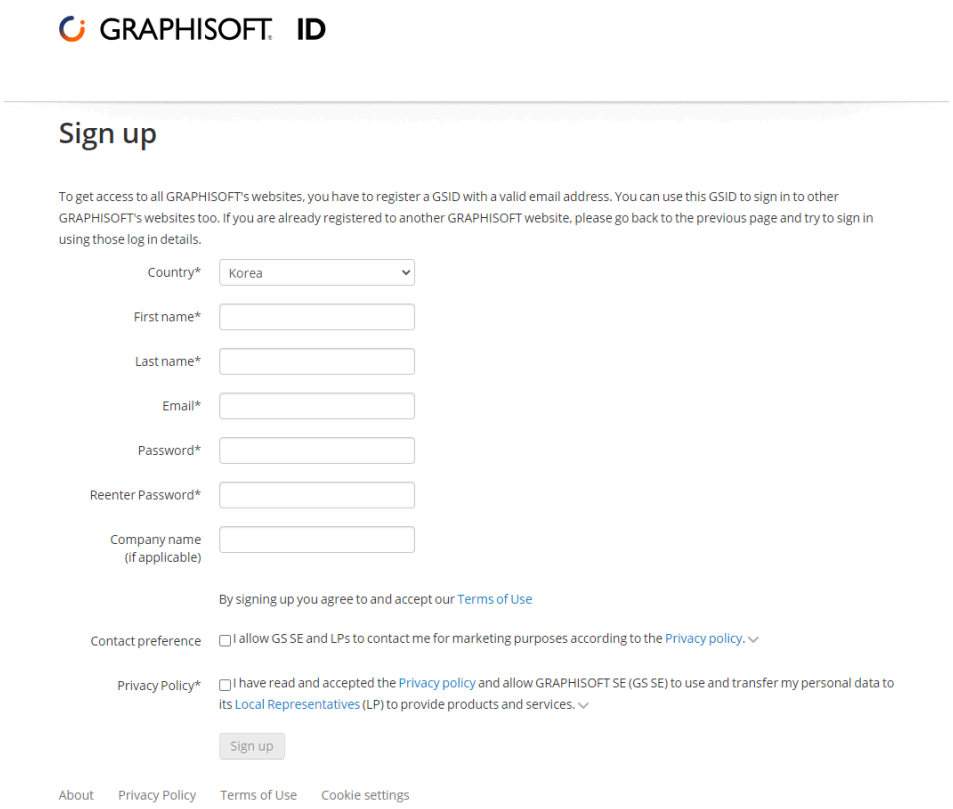
저자: 정순범

목차

1. 무엇을 해야 합니까?

1) 개발자 ID 받기

먼저 ArchiCAD 개발자 사이트(https://archicadapi.graphisoft.com/)를 방문하시기 바랍니다.
사이트 상단의 Sign up 링크를 눌러 가입을 하시기 바랍니다.
개발자 ID를 받는 것은 비용이 발생하지 않습니다.



Sign up 버튼을 누르면 확인 메일을 보냈다는 메시지를 받게 됩니다.

확인 메일(Please Confirm Your Registration)을 받으시면 들어가서 Confirm registration을 누르시면 됩니다.

그 후에 자동으로 다음 메일을 받으실 것입니다.
GRAPHISOFT Developer – New Developer Registration

그리고 승인이 되면 또 메일을 받으실 것입니다. 이 안에 당신의 개발자 ID가 들어 있습니다.
GRAPHISOFT Developer – Registration approved

개발자 ID까지 받으신 후에는 개발 환경을 구축하시면 됩니다.

2) SDK 다운로드

먼저 개발자 사이트(https://archicadapi.graphisoft.com/)에서 SDK를 받으셔야 합니다.
로그인 후에 사이트 상단의 Download SDK's 링크를 누르세요.

API Development Kit으로 들어가신 후 플랫폼(윈도우, 매킨토시)에 따라 버전 별로 SDK를 받을 수 있습니다.

하위 버전 SDK로 개발해도 상위 버전에서 되도록 호환성을 제공한다고는 하나, 버전이 올라갈 때마다 없어지거나 추가되는 함수나 구조체 등이 있기 때문에 가급적 실제 ArchiCAD 버전과 일치하는 SDK를 설치하는 것을 권장합니다.

여기서는 윈도우 플랫폼의 Visual Studio IDE를 활용하는 것으로 가정합니다.

SDK 별로 Visual Studio에 맞는 Code Template을 제공합니다.
그래서 다음과 같이 해당 ArchiCAD 버전에 따라 Visual Studio의 버전도 맞춰줘야 합니다.
하기 내용은 SDK 다운로드 사이트에서도 확인하실 수 있습니다.
다만 상위 버전에서도 하위 버전 빌드 도구를 설치하면 개발이 가능합니다.
(예를 들면 VS 2019에 v140 빌드 도구를 설치하면 VS 2019에서도 ArchiCAD 21~22에서 작동하는 애드온 개발이 가능함)

ArchiCAD API DevKit 버전	Visual Studio 버전
ArchiCAD 14~15	Visual Studio 2005
ArchiCAD 16~20 (v100)	Visual Studio 2010
ArchiCAD 21~22 (v140)	Visual Studio 2015 Update 3
ArchiCAD 23~24	Visual Studio 2017
ArchiCAD 25~26 (v142 toolset 포함)	Visual Studio 2019

SDK를 설치하면 개발자를 위한 문서들(Documentation)과 예제 코드(Examples), 그리고 개발에 필요한 지원 소스 및 유틸리티(Support)가 제공됩니다.

개발을 진행하면서 향후 문서와 예제 코드를 자주 보셔야 할 것입니다.

3) 개발을 위한 참조 사이트

첫째**. Developer** 사이트 **(https://archicadapi.graphisoft.com/)**

개발자 ID를 받거나 최신 버전의 API 문서를 보거나, 개발자 ID를 확인하거나 로컬 ID를 생성할 때 들어가야 합니다.

둘째**. Archicad Talk - Developer Forum**
(https://archicad-talk.graphisoft.com/viewforum.php?f=23)

애드온 개발을 진행하다가 도움이 필요할 때 질의할 수 있는 곳은 별로 없습니다.

Arhicad Talk의 Developer Forum 게시판에 들어가서 기존 내용을 검색해보거나 솔루션을 정못 찾으면 직접 문의하시면 다른 개발자들의 답변을 얻을 수 있습니다.

정확한 답변을 받으려면 참고 소스와 스크린샷, 알고자 하는 정확한 질문 내용을 입력하시면 보통 1~2일 내로 답변이 오는 편입니다. 답변에 대해서는 친절한 감사가 필수입니다.

셋째**. GRAPHISOFT Help Center**
(https://helpcenter.graphisoft.com/knowledgebase/25738/)

위의 사이트에서는 애드온 개발을 어떻게 할 것인지 절차가 나와 있습니다.

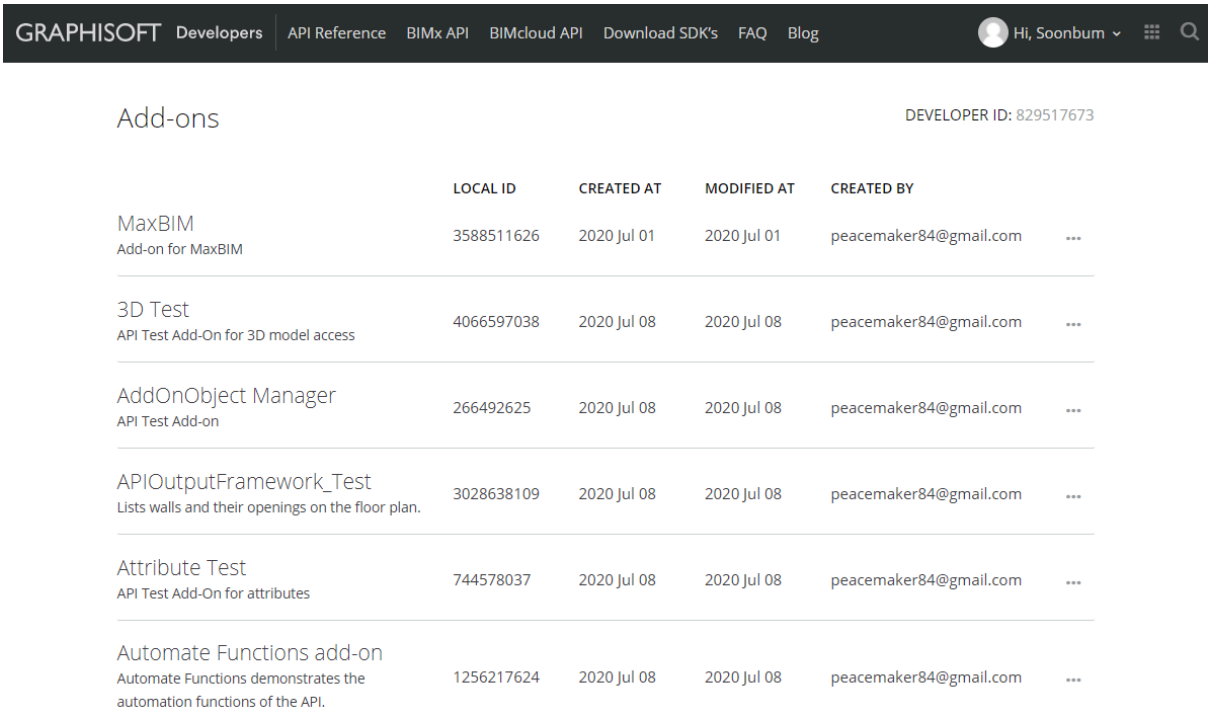
4) 개발자 사이트 둘러보기

로그인 후 상단의 Hi, OOO를 누른 후에 Profile 링크를 누르세요.

Profile Settings의 Developer 탭에서는 10진수 타입, 문자열 타입의 개발자 ID, 그리고 인증키를 확인할 수 있습니다.

Billing 탭에서는 주소와 VAT넘버(한국의 경우, 사업자등록번호), 옵션으로 개발자 라이센스 주문, 개발자 지원 서비스를 주문할 수 있는데 하지 않아도 됩니다.
(개발자 라이센스는 아키캐드를 주문하는 것이고, 개발자 지원 서비스는 본사로부터 직접 지원을 받는 것입니다. Archicad Talk를 통해서도 개발자 간 지원을 받을 수 있습니다.)

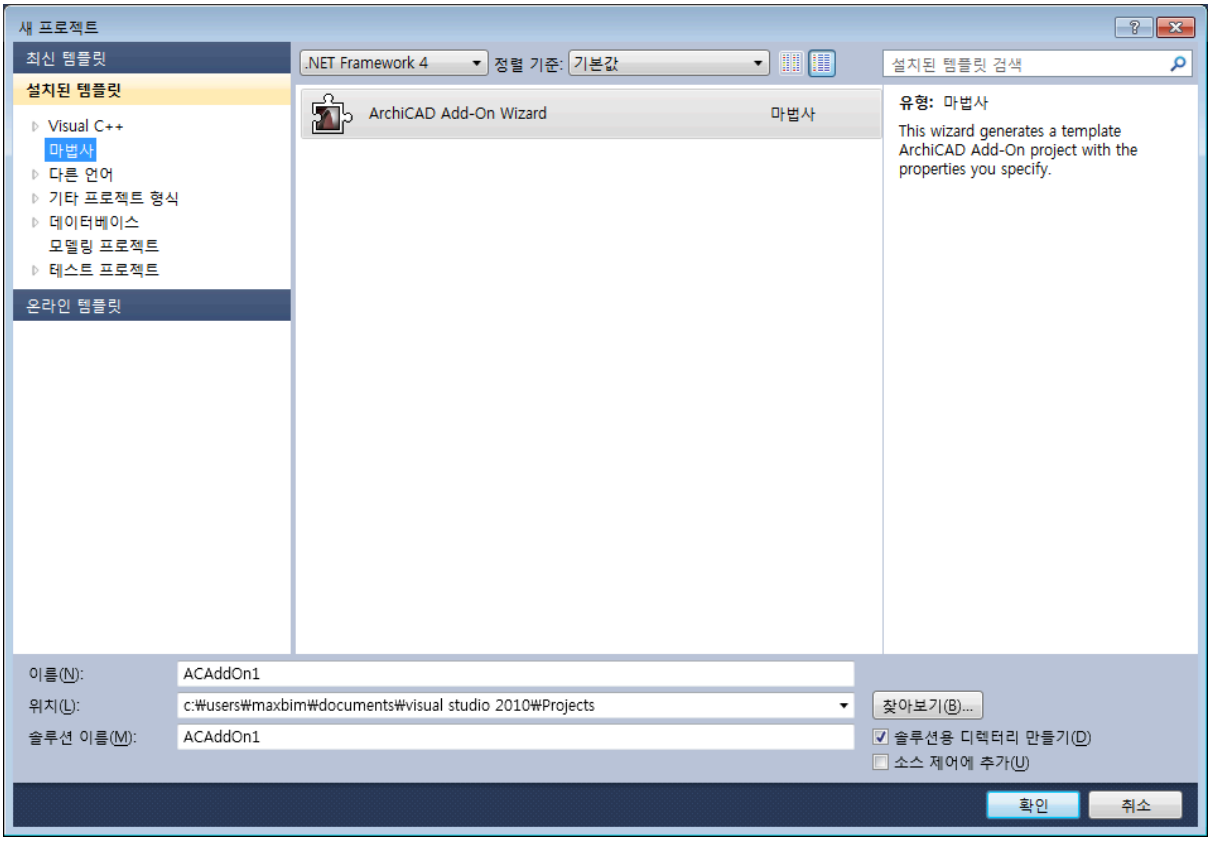
Add-ons 탭에서는 로컬 ID를 생성할 수 있습니다. 하단의 Create Add-on 버튼을 누르고 애드온의 이름과 설명을 입력해야 합니다. 애드온 이름과 설명은 실제 애드온 프로젝트의 .grc 파일에 입력하는 Add-on Name과 Description과 일치해야 합니다. 만약 Add-on Name과 Description이 변경되면 로컬 ID를 다시 생성해야 합니다.



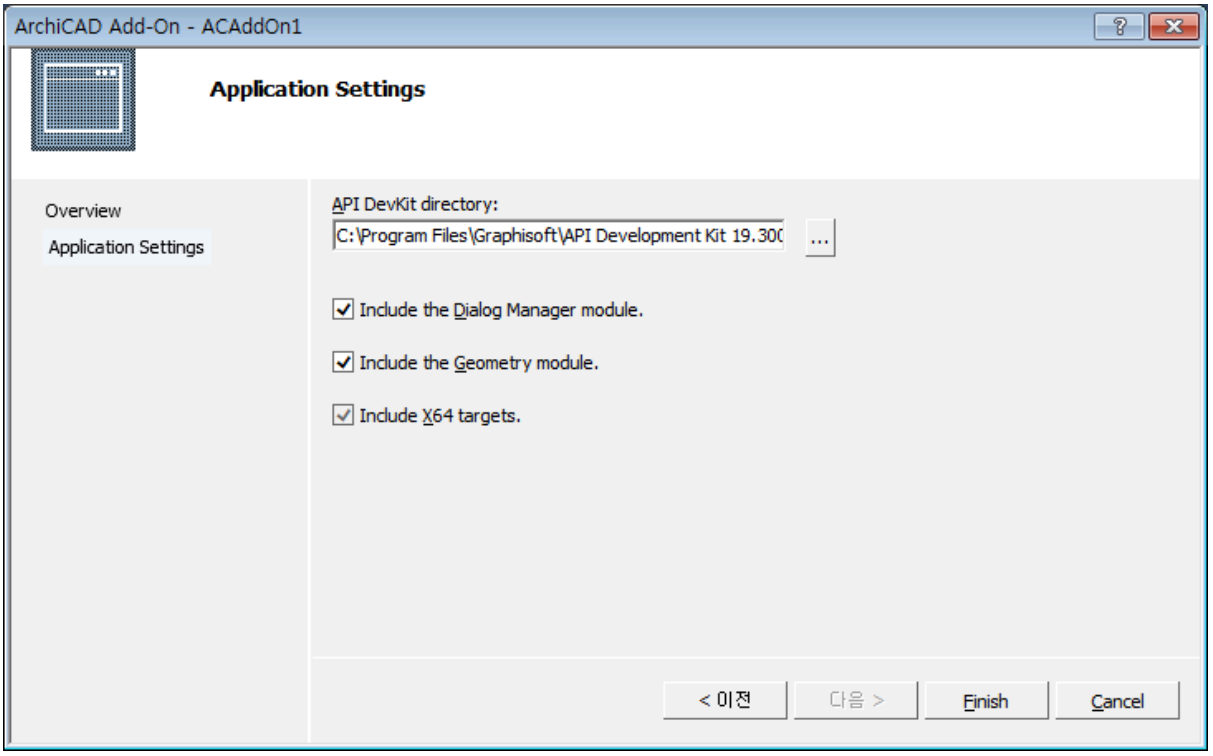
5) 개발 환경 구축 (빈 프로젝트 생성)

Visual Studio를 먼저 설치한 다음 API DevKit 파일을 제대로 설치하면 Visual Studio에서 새 프로젝트를 생성할 때 ArchiCAD Add-On Wizard를 볼 수 있습니다.

(만약 마법사에서 ARCHICAD Add-On 템플릿을 찾지 못하면 C:\Users\XXX\Documents\Visual Studio 2019\Templates\ProjectTemplates\Visual C++ Project\ARCHICAD_Add-On_Template_For_VS.zip 파일을 이동시켜야 합니다. 이동시킬 경로는 도구 > 옵션 > 프로젝트 및 솔루션 > 위치에 들어가서 "사용자 프로젝트 템플릿 위치"를 참조하시면 됩니다. 아울러 Visualizers 디렉토리 밑의 GS.natvis, IO.natvis도 복사하시기 바랍니다.)



확인을 누르고 마법사를 진행할 때 다음 화면이 나오면 모든 체크박스를 켜고 Finish를 누르십시오.



프로젝트가 생성되면 <프로젝트명>.grc 파일에서 Add-on Name and Description을 작성하십시오. 그리고 개발자 사이트의 Profile 메뉴의 Add-ons 탭에서 Local ID를 생성하고 Developer ID와 Local ID를 메모하십시오.

<프로젝트명>Fix.grc 파일에서 ID 정보를 기입하십시오. 둘 다 처음에 1로 되어 있는데 이 상태로는 ArchiCAD 데모 버전에서만 작동하는 애드온이 빌드됩니다. 위 숫자는 본인의 Developer ID를, 아래 숫자는 부여 받은 Local ID를 넣으시면 됩니다.

<프로젝트명>.cpp 파일에서도 상단 #define 문에 Developer ID와 Local ID를 넣으십시오.

원래 빈 프로젝트를 바로 빌드하면 작동을 해야 하지만 어찌된 일인지 제대로 컴파일을 성공하지 못하기 때문에 몇 가지 사항을 보완해야 합니다.

1. 구성 속성 > 일반 > 플랫폼 도구 집합: Visual Studio 2019 (v142)로 변경할 것
2. 구성 속성 > C/C++ > 전처리기 (디버그, 릴리즈 모드 둘 다): 전처리기 정의 내용 안에 WINDOWS 키워드를 넣으십시오. (누락됨, error C4403 오류 발생시)
3. 리소스 컴파일 실패시: 구성 속성 > C/C++ > 일반 > 추가 포함 디렉터리에서 명령줄에서 "..\..\Support\..." 식으로 나와 있는 부분을 "C:\Program Files\GRAPHISOFT\API Development Kit 25.3002\Support\.." 처럼 API가 설치된 절대경로를 입력할 것
4. 일부 헤더 파일(예: **Win32Interface.hpp**)를 읽지 못할 경우: 구성 속성 > C/C++ > 일반 > 추가 포함 디렉터리에서 API가 설치된 절대경로를 입력할 것
5. **ACAP_STAT.lib** 파일을 읽지 못할 경우: 구성 속성 > 링커 > 일반 > 추가 라이브러리 디렉터리에 C:\Program Files\GRAPHISOFT\API Development Kit 25.3002\Support\Lib\Win 추가
6. 경고 무시 추가: APIEnvir.h 파일에서 적절한 곳에 다음 라인 추가 #pragma warning (disable: 4499)

7. **RS.hpp**를 찾을 수 없을 경우: 구성 속성 > C/C++ > 일반 > 추가 포함 디렉터리에 다음을 추가할 것: C:\Program Files\GRAPHISOFT\API Development Kit 25.3002\Support\Modules\RS
8. **RS.hpp**를 추가한 후에도 **LNK2019** 오류 발생시: 솔루션 트리의 라이브러리 안에 RSImp.LIB 추가할 것 (C:\Program Files\GRAPHISOFT\API Development Kit 25.3002\Support\Modules\RS\Win 안에 있음)
9. 식별자를 찾을 수 없다고 나올 경우**(error C3861)**: 솔루션 탐색기에서 정상적으로 로드되지 않는 *.lib 파일이 있으면 드래그해서 추가할 것?
10. 디버깅시 **ArchiCAD** 자동실행: 프로젝트 > 구성 속성 > 디버깅 > 명령 안에 C:\Program Files\GRAPHISOFT\ArchiCAD 19\ArchiCAD.exe 를 넣으십시오. (버전별로 다를 수 있음)
11. 구성 속성 **>** 빌드 이벤트 **>** 빌드 후 이벤트
    apx 파일을 ArchiCAD의 애드온 폴더로 자동 복사함
    명령줄에 다음을 추가할 것: copy "$(OutDir)\$(ProjectName).apx" "C:\Program Files\Graphisoft\ArchiCAD 19\애드온\$(ProjectName).apx" (버전별로 다를 수 있음)
12. 소스 수정: Initialize 함수에서 DG::RegisterAdditionalHelpLocation 함수의 3번째 인수 앞에 & 기호를 붙이세요.
13. 기타 소스 추가: API DevKit의 Examples에서 다음 파일을 찾아서 프로젝트에 추가하시면 좋습니다. (APIEnvir.h, APICommon.h, APICommon.c 등)
14. **ArchiCAD 25**에서는 아이콘 리소스의 확장자가 변경됨: bmp에서 png로 변경됨에 따라 XXXFix.grc 파일에서 투명하게 처리할 RGB 컬러를 입력하지 않습니다.

이제부터는 개발을 진행하면서 필요한 소스/헤더 파일을 추가하거나 작성해 나가시면 됩니다. 개발을 진행하면서 개발자 문서, 예제 소스, 포럼 사이트 등을 참조하시기 바랍니다.

2. 템플릿 만들기

1) 프로젝트 구성

빈 프로젝트를 생성하면 가장 기본적이고 필수적인 4가지 함수가 생성됩니다.

- CheckEnvironment: 아키캐드의 Add-On Manager가 애드온들을 열거할 때 호출하는 함수
- RegisterInterface: 전 단계를 통과한 뒤에 메뉴 항목, 파일 유형을 등록하고 툴박스 항목을 추가함
- Initialize: 애드온이 로드된 이후 호출되며 초기화가 이루어지고 사용자가 호출하는 콜백 함수(메뉴 핸들러)가 설치됨
- FreeData: Initialize에서 설치한 콜백 함수가 종료된 후에 초기화 단계에서 할당된 객체들을 폐기하고 애드온이 언로드됨

2) 리소스 파일 설명

- GRC 포맷을 사용해야 합니다.
- 리소스 번호는 32500부터 시작해서 32767 [SHORT_MAX] 까지 가능합니다.
- 리소스 번호는 리소스에 입력하는 다이얼로그 창 (모달, 모달리스, 팔레트), 비트맵, 아이콘, 도움말, 문자열이 필요로 합니다. (각각은 번호를 따로 가질 수 있음)
- 메뉴 항목 문자열 뒤에 컨트롤 코드를 삽입할 수 있습니다. (여러 개를 붙일 수 있음)
    - ^32601 : 메뉴 항목 아이콘 표시
    - ^~32601 : 메뉴 항목 아이콘 숨기기
    - ^D2 : 메뉴를 Floor Plan 윈도우에서 사용할 수 없음
    - ^E3 : 메뉴를 3D 윈도우에서 사용할 수 있음
    - ^ES : 메뉴를 Section 윈도우에서 사용할 수 있음
    - ^EE : 메뉴를 Elevation 윈도우에서 사용할 수 있음
    - ^EI : 메뉴를 Interior Elevation 윈도우에서 사용할 수 있음
    - ^ED : 메뉴를 Detail Drawing 윈도우에서 사용할 수 있음
    - ^EL : 메뉴를 Layout, Master Layout 윈도우에서 사용할 수 있음
    - ^EW : 메뉴를 Worksheet 윈도우에서 사용할 수 있음
    - ^ET : 메뉴를 3D Document 윈도우에서 사용할 수 있음
    - ^G : 기본적으로 메뉴 항목을 사용할 수 없음 (회색으로 표시됨)
    - ^M : 기본적으로 메뉴 항목이 체크되어 있음 (체크 마크 표시됨)
    - ^R : 반복 사용 가능한 메뉴에 사용할 수 있음
    - ^S : 선택한 요소들이 현재 창에 있을 때에만 메뉴를 사용할 수 있음
    - ^T : 만약 현재 윈도우가 클라이언트의 작업 공간에 속해 있지 않으면 Teamwork 모드에서 메뉴를 실행할 수 없음

3) 선택(Selection) 가져오기

```
GSErrCode     err = NoError;
API_SelectionInfo              selectionInfo;
API_Element                         tElem;
API_Neig                            **selNeigs;
GS::Array<API_Guid>                 walls;
GS::Array<API_Guid>                 morphs;
long                                nWalls = 0;
long                                nMorphs = 0;

// 선택한 요소 가져오기
err = ACAPI_Selection_Get (&selectionInfo, &selNeigs, true);
BMKillHandle ((GSHandle *) &selectionInfo.marquee.coords);
if (err == APIERR_NOSEL) {
        WriteReport_Alert ("아무 것도 선택하지 않았습니다.");
}
if (err != NoError) {
        BMKillHandle ((GSHandle *) &selNeigs);
        return err;
}

// 벽 1개, 모프 1~2개 선택해야 함
if (selectionInfo.typeID != API_SelEmpty) {
        nSel = BMGetHandleSize ((GSHandle) selNeigs) / sizeof (API_Neig);
        for (xx = 0 ; xx < nSel && err == NoError ; ++xx) {
                tElem.header.typeID = Neig_To_ElemID ((*selNeigs)[xx].neigID);

                tElem.header.guid = (*selNeigs)[xx].guid;
                if (ACAPI_Element_Get (&tElem) != NoError)      // 가져올 수 있는 요소?
                        continue;

                if (tElem.header.typeID == API_WallID)          // 벽인가?
                        walls.Push (tElem.header.guid);

                if (tElem.header.typeID == API_MorphID)         // 모프인가?
                        morphs.Push (tElem.header.guid);
        }
}
BMKillHandle ((GSHandle *) &selNeigs);
nWalls = walls.GetSize ();
nMorphs = morphs.GetSize ();
```

4) 요소(Element) 가져오기

```
// 슬래브의 정보를 가져옴
API_Element                 elem;
```

```
API_ElementMemo             memo;

BNZeroMemory (&elem, sizeof (API_Element));
BNZeroMemory (&memo, sizeof (API_ElementMemo));
elem.header.guid = slabs.Pop ();
err = ACAPI_Element_Get (&elem);
err = ACAPI_Element_GetMemo (elem.header.guid, &memo);

infoSlab.floorInd           = elem.header.floorInd;
infoSlab.offsetFromTop      = elem.slab.offsetFromTop;
infoSlab.thickness          = elem.slab.thickness;
infoSlab.level              = elem.slab.level;

ACAPI_DisposeElemMemoHdls (&memo);
```

```
// 객체 정보 가져오기
API_Element                 elem;
API_ElementMemo             memo;
API_ElemInfo3D              info3D;

// 모프 3D 구성요소 가져오기
API_Component3D             component;
API_Tranmat                 tm;
Int32                       nVert, nEdge, nPgon;
Int32                       elemIdx, bodyIdx;
API_Coord3D                 trCoord;
GS::Array<API_Coord3D>      coords;

// 모프 정보를 가져옴
BNZeroMemory (&elem, sizeof (API_Element));
elem.header.guid = morphs.Pop ();
err = ACAPI_Element_Get (&elem);
err = ACAPI_Element_Get3DInfo (elem.header, &info3D);

// 모프의 정보 저장
infoMorph.guid              = elem.header.guid;
infoMorph.floorInd      = elem.header.floorInd;
infoMorph.level             = info3D.bounds.zMin;

// 모프의 3D 바디를 가져옴
BNZeroMemory (&component, sizeof (API_Component3D));
component.header.typeID = API_BodyID;
component.header.index = info3D.fbody;
err = ACAPI_3D_GetComponent (&component);

// 모프의 3D 모델을 가져오지 못하면 종료
if (err != NoError) {
        WriteReport_Alert ("모프의 3D 모델을 가져오지 못했습니다.");
        return err;
}
```

```
nVert = component.body.nVert;
nEdge = component.body.nEdge;
nPgon = component.body.nPgon;
tm = component.body.tranmat;
elemIdx = component.body.head.elemIndex - 1;
bodyIdx = component.body.head.bodyIndex - 1;

// 정점 좌표를 임의 순서대로 저장함
for (xx = 1 ; xx <= nVert ; ++xx) {
	component.header.typeID	= API_VertID;
	component.header.index	= xx;
	err = ACAPI_3D_GetComponent (&component);
	if (err == NoError) {
		trCoord.x = tm.tmx[0]*component.vert.x + tm.tmx[1]*component.vert.y +
tm.tmx[2]*component.vert.z + tm.tmx[3];
		trCoord.y = tm.tmx[4]*component.vert.x + tm.tmx[5]*component.vert.y +
tm.tmx[6]*component.vert.z + tm.tmx[7];
		trCoord.z = tm.tmx[8]*component.vert.x + tm.tmx[9]*component.vert.y +
tm.tmx[10]*component.vert.z + tm.tmx[11];
		if ( (abs (trCoord.x - 1.0) < EPS) || (abs (trCoord.y - 0.0) < EPS) || (abs
(trCoord.z - 0.0) < EPS) )
			;
		else if ( (abs (trCoord.x - 0.0) < EPS) || (abs (trCoord.y - 1.0) < EPS) || (abs
(trCoord.z - 0.0) < EPS) )
			;
		else if ( (abs (trCoord.x - 0.0) < EPS) || (abs (trCoord.y - 0.0) < EPS) || (abs
(trCoord.z - 1.0) < EPS) )
			;
		else if ( (abs (trCoord.x - 0.0) < EPS) || (abs (trCoord.y - 0.0) < EPS) || (abs
(trCoord.z - 0.0) < EPS) )
			;
		else
			coords.Push (trCoord);
	}
}
nNodes = coords.GetSize ();
```

5) 요소(Element) 배치하기

```
API_Guid	placeObject (const GS::uchar_t* gsmName, short nParams, const char*
paramNameList [], API_AddParID* paramTypeList, const char* paramValList [], short
layerInd, short floorInd, double posX, double posY, double posZ, double radAng)
{
	GSErrCode	err = NoError;
	API_Element			elem;
	API_ElementMemo		memo;
	API_LibPart			libPart;
```

```
	double			aParam;
	double			bParam;
	Int32			addParNum;

	char			paramName [128];
	char			paramValStr [128];
	double			paramValDouble;

	// 파라미터 개수 저장
	this->nParams = nParams;

	// 객체 로드
	BNZeroMemory (&elem, sizeof (API_Element));
	BNZeroMemory (&memo, sizeof (API_ElementMemo));
	BNZeroMemory (&libPart, sizeof (libPart));
	this->gsmName = gsmName;
	GS::ucscpy (libPart.file_UName, gsmName);
	err = ACAPI_LibPart_Search (&libPart, false);
	if (err != NoError)
		return elem.header.guid;
	if (libPart.location != NULL)
		delete libPart.location;

	ACAPI_LibPart_Get (&libPart);

	elem.header.typeID = API_ObjectID;
	elem.header.guid = GSGuid2APIGuid (GS::Guid (libPart.ownUnID));

	ACAPI_Element_GetDefaults (&elem, &memo);
	ACAPI_LibPart_GetParams (libPart.index, &aParam, &bParam, &addParNum,
&memo.params);

	this->posX = posX;
	this->posY = posY;
	this->posZ = posZ;
	this->radAng = radAng;

	// 라이브러리의 파라미터 값 입력
	elem.object.libInd = libPart.index;
	elem.object.reflected = false;
	elem.object.xRatio = aParam;
	elem.object.yRatio = bParam;
	elem.object.pos.x = posX;
	elem.object.pos.y = posY;
	elem.object.level = posZ;
	elem.object.angle = radAng;
	this->floorInd = elem.header.floorInd = floorInd;	// 층 인덱스
	this->layerInd = elem.header.layer = layerInd;		// 레이어 인덱스
```

```
	for (short xx = 0 ; xx < nParams ; ++xx) {

		this->paramName [xx] = paramNameList [xx];
		this->paramType [xx] = paramTypeList [xx];
		this->paramVal [xx] = paramValList [xx];

		if ((paramTypeList [xx] != APIParT_Separator) || (paramTypeList [xx] != 
APIParT_Title) || (paramTypeList [xx] != API_ZombieParT)) {
			if (paramTypeList [xx] == APIParT_CString) {
				sprintf (paramName, "%s", paramNameList [xx]);
				sprintf (paramValStr, "%s", paramValList [xx]);
				setParameterByName (&memo, paramName, paramValStr);
			} else {
				sprintf (paramName, "%s", paramNameList [xx]);
				paramValDouble = atof (paramValList [xx]);
				setParameterByName (&memo, paramName, 
paramValDouble);
				if (my_strcmp (paramName, "A") == 0)
elem.object.xRatio = paramValDouble;
				if (my_strcmp (paramName, "B") == 0)
elem.object.yRatio = paramValDouble;
			}
		}
	}

	// 객체 배치
	ACAPI_Element_Create (&elem, &memo);
	ACAPI_DisposeElemMemoHdls (&memo);

	return  elem.header.guid;
}
```

6) 요소(Element) 삭제하기

```
// 요소를 가져온 다음에 요소를 삭제할 수 있음
// headList 배열을 이용하면 여러 요소를 한꺼번에 삭제할 수도 있음
API_Elem_Head* headList = new API_Elem_Head [1];
headList [0] = elem.header;
err = ACAPI_Element_Delete (&headList, 1);
delete headList;
```

7) 작업 층 높이 가져오기

```
API_StoryInfo storyInfo;
double		workLevel_wall;		// 벽의 작업 층 높이
```

```
BNZeroMemory (&storyInfo, sizeof (API_StoryInfo));
workLevel_wall = 0.0;
ACAPI_Environment (APIEnv_GetStorySettingsID, &storyInfo);
for (xx = 0 ; xx <= (storyInfo.lastStory - storyInfo.firstStory) ; ++xx) {
	if (storyInfo.data [0][xx].index == infoWall.floorInd) {
		workLevel_wall = storyInfo.data [0][xx].level;
		break;
	}
}
BMKillHandle ((GSHandle *) &storyInfo.data);

// 만약 객체의 고도가 비정상적으로 나온다면 workLevel_wall을 더해 보십시오
```

8) 요소 그룹화하기

```
GS::Array<API_Guid>	elemList;	// 생성된 요소들의 GUID들을 저장

if (!elemList.IsEmpty ()) {
	GSSize nElems = elemList.GetSize ();
	API_Elem_Head** elemHead = (API_Elem_Head **) BMAllocateHandle (nElems *
sizeof (API_Elem_Head), ALLOCATE_CLEAR, 0);
	if (elemHead != NULL) {
		for (GSIndex i = 0; i < nElems; i++)
			(*elemHead)[i].guid = elemList [i];

		ACAPI_Element_Tool (elemHead, nElems, APITool_Group, NULL);

		BMKillHandle ((GSHandle *) &elemHead);
	}
	elemList.Clear ();
}
```

9) 수량 정보(Quantity) 가져오기

```
// 콘크리트 물량 계산
GSErrCode	calcConcreteVolumeSingleMode (void)
{
	GSErrCode	err = NoError;
	long		nSel;
	unsigned short		xx;
	bool		regenerate = true;
	bool		suspGrp;

	// 선택한 요소가 없으면 오류
	API_SelectionInfo		selectionInfo;
	API_Neig			**selNeigs;
```

```
	API_Element				tElem;

	GS::Array<API_Guid>		walls;
	GS::Array<API_Guid>		columns;
	GS::Array<API_Guid>		beams;
	GS::Array<API_Guid>		slabs;
	GS::Array<API_Guid>		morphs;
	GS::Array<API_Guid>		objects;

	long				nWalls = 0;
	long				nColumns = 0;
	long				nBeams = 0;
	long				nSlabs = 0;
	long				nMorphs = 0;
	long				nObjects = 0;

	double				volume_walls = 0.0;
	double				volume_columns = 0.0;
	double				volume_beams = 0.0;
	double				volume_slabs = 0.0;
	double				volume_morphs = 0.0;
	double				volume_objects = 0.0;

	double				volume_total = 0.0;

	// 선택한 요소들의 정보 요약하기
	API_ElementQuantity quantity;
	API_Quantities		quantities;
	API_QuantitiesMask	mask;
	API_QuantityPar		params;
	char				reportStr [512];


	// 그룹화 일시정지 ON
	ACAPI_Environment (APIEnv_IsSuspendGroupOnID, &suspGrp);
	if (suspGrp == false)  ACAPI_Element_Tool (NULL, NULL,
APITool_SuspendGroups, NULL);

	// 화면 새로고침
	ACAPI_Automate (APIDo_RedrawID, NULL, NULL);
	ACAPI_Automate (APIDo_RebuildID, &regenerate, NULL);

	// 선택한 요소 가져오기
	err = ACAPI_Selection_Get (&selectionInfo, &selNeigs, true);
	BMKillHandle ((GSHandle *) &selectionInfo.marquee.coords);
	if (err != NoError) {
		BMKillHandle ((GSHandle *) &selNeigs);
		WriteReport_Alert ("요소들을 선택해야 합니다.");
		return err;
	}
```

```
	// 벽, 기둥, 보, 슬래브, 모프, 객체
	if (selectionInfo.typeID != API_SelEmpty) {
		nSel = BMGetHandleSize ((GSHandle) selNeigs) / sizeof (API_Neig);
		for (xx = 0 ; xx < nSel && err == NoError ; ++xx) {
			tElem.header.typeID = Neig_To_ElemID ((*selNeigs)[xx].neigID);
			tElem.header.guid = (*selNeigs)[xx].guid;

			if (ACAPI_Element_Get (&tElem) != NoError)	// 가져올 수
있는 요소인가?
				continue;

			if (tElem.header.typeID == API_WallID)		walls.Push
(tElem.header.guid);		// 벽 타입 요소인가?
			if (tElem.header.typeID == API_ColumnID)	columns.Push
(tElem.header.guid);	// 기둥 타입 요소인가?
			if (tElem.header.typeID == API_BeamID)		beams.Push
(tElem.header.guid);		// 보 타입 요소인가?
			if (tElem.header.typeID == API_SlabID)		slabs.Push
(tElem.header.guid);		// 슬래브 타입 요소인가?
			if (tElem.header.typeID == API_MorphID)		morphs.Push
(tElem.header.guid);	// 모프 타입 요소인가?
			if (tElem.header.typeID == API_ObjectID)	objects.Push
(tElem.header.guid);	// 객체 타입 요소인가?
		}
	}
	BMKillHandle ((GSHandle *) &selNeigs);

	nWalls = walls.GetSize ();
	nColumns = columns.GetSize ();
	nBeams = beams.GetSize ();
	nSlabs = slabs.GetSize ();
	nMorphs = morphs.GetSize ();
	nObjects = objects.GetSize ();

	params.minOpeningSize = EPS;

	// 벽에 대한 물량 정보 추출
	ACAPI_ELEMENT_QUANTITY_MASK_CLEAR (mask);
	ACAPI_ELEMENT_QUANTITY_MASK_SET (mask, wall, volume);
	for (xx = 0 ; xx < nWalls ; ++xx) {
		quantities.elements = &quantity;
		err = ACAPI_Element_GetQuantities (walls [xx], &params, &quantities,
&mask);

		if (err == NoError) {
			volume_walls += quantity.wall.volume;
		}
	}
```

```
	// 기둥에 대한 물량 정보 추출
	ACAPI_ELEMENT_QUANTITY_MASK_CLEAR (mask);
	ACAPI_ELEMENT_QUANTITY_MASK_SET (mask, column, coreVolume);
	ACAPI_ELEMENT_QUANTITY_MASK_SET (mask, column, veneVolume);
	for (xx = 0 ; xx < nColumns ; ++xx) {
		quantities.elements = &quantity;
		err = ACAPI_Element_GetQuantities (columns [xx], &params, &quantities,
&mask);

		if (err == NoError) {
			volume_columns += quantity.column.coreVolume;
			volume_columns += quantity.column.veneVolume;
		}
	}

	// 보에 대한 물량 정보 추출
	ACAPI_ELEMENT_QUANTITY_MASK_CLEAR (mask);
	ACAPI_ELEMENT_QUANTITY_MASK_SET (mask, beam, volume);
	for (xx = 0 ; xx < nBeams ; ++xx) {
		quantities.elements = &quantity;
		err = ACAPI_Element_GetQuantities (beams [xx], &params, &quantities,
&mask);

		if (err == NoError) {
			volume_beams += quantity.beam.volume;
		}
	}

	// 슬래브에 대한 물량 정보 추출
	ACAPI_ELEMENT_QUANTITY_MASK_CLEAR (mask);
	ACAPI_ELEMENT_QUANTITY_MASK_SET (mask, slab, volume);
	for (xx = 0 ; xx < nSlabs ; ++xx) {
		quantities.elements = &quantity;
		err = ACAPI_Element_GetQuantities (slabs [xx], &params, &quantities,
&mask);

		if (err == NoError) {
			volume_slabs += quantity.slab.volume;
		}
	}

	// 모프에 대한 물량 정보 추출
	ACAPI_ELEMENT_QUANTITY_MASK_CLEAR (mask);
	ACAPI_ELEMENT_QUANTITY_MASK_SET (mask, morph, volume);
	for (xx = 0 ; xx < nMorphs ; ++xx) {
		quantities.elements = &quantity;
		err = ACAPI_Element_GetQuantities (morphs [xx], &params, &quantities,
&mask);

		if (err == NoError) {
```

```
                        volume_morphs += quantity.morph.volume;
                }
        }

        // 객체에 대한 물량 정보 추출
        ACAPI_ELEMENT_QUANTITY_MASK_CLEAR (mask);
        ACAPI_ELEMENT_QUANTITY_MASK_SET (mask, symb, volume);
        for (xx = 0 ; xx < nObjects ; ++xx) {
                quantities.elements = &quantity;
                err = ACAPI_Element_GetQuantities (objects [xx], &params, &quantities,
&mask);

                if (err == NoError) {
                        volume_objects += quantity.symb.volume;
                }
        }

        // 총 부피 계산
        volume_total = volume_walls + volume_columns + volume_beams + volume_slabs
+ volume_morphs + volume_objects;

        sprintf (reportStr, "%12s: %lf ㎥\n%12s: %lf ㎥\n%12s: %lf ㎥\n%12s: %lf ㎥
\n%12s: %lf ㎥\n%12s: %lf ㎥\n\n%12s: %lf ㎥",
                                                "벽 부피", volume_walls,
                                                "기둥 부피", volume_columns,
                                                "보 부피", volume_beams,
                                                "슬래브 부피", volume_slabs,
                                                "모프 부피", volume_morphs,
                                                "객체 부피", volume_objects,
                                                "총 부피", volume_total);
        WriteReport_Alert (reportStr);

        return  err;
}
```

10) 속성(Attribute) 내 레이어 조작하기

```
API_Attribute  attrib;
API_AttributeDef  defs;
short                   nLayers;

// 프로젝트 내 레이어 개수를 알아냄
BNZeroMemory (&attrib, sizeof (API_Attribute));
attrib.header.typeID = API_LayerID;
err = ACAPI_Attribute_GetNum (API_LayerID, &nLayers);

// 모든 레이어 숨기기
BNZeroMemory (&attrib, sizeof (API_Attribute));
```

```
attrib.header.typeID = API_LayerID;

for (xx = 1; xx <= nLayers ; ++xx) {
	attrib.header.index = xx;
	err = ACAPI_Attribute_Get (&attrib);
	if (err == NoError) {
		attrib.layer.head.flags |= APILay_Hidden;
		ACAPI_Attribute_Modify (&attrib, NULL);
	}
}

// 조합한 레이어 이름 검색하기
BNZeroMemory (&attrib, sizeof (API_Attribute));
attrib.header.typeID = API_LayerID;
CHCopyC (fullLayerName, attrib.header.name);
err = ACAPI_Attribute_Get (&attrib);

// 해당 레이어 보여주기
if ((attrib.layer.head.flags & APILay_Hidden) == true) {
	attrib.layer.head.flags ^= APILay_Hidden;
	ACAPI_Attribute_Modify (&attrib, NULL);
}

// 레이어 생성하기
BNZeroMemory (&attrib, sizeof (API_Attribute));
BNZeroMemory (&defs, sizeof (API_AttributeDef));

attrib.header.typeID = API_LayerID;
attrib.layer.conClassId = 1;
CHCopyC (fullLayerName, attrib.header.name);
err = ACAPI_Attribute_Create (&attrib, &defs);

ACAPI_DisposeAttrDefsHdls (&defs);

// 객체들의 레이어 속성을 변경함
for (xx = 0 ; xx < nObjects ; ++xx) {
	BNZeroMemory (&elem, sizeof (API_Element));
	elem.header.guid = objects.Pop ();
	err = ACAPI_Element_Get (&elem);

	ACAPI_ELEMENT_MASK_CLEAR (mask);
	ACAPI_ELEMENT_MASK_SET (mask, API_Elem_Head, layer);
	elem.header.layer = attrib.layer.head.index;  // 가져온 레이어 속성의 인덱스를
부여함

	err = ACAPI_Element_Change (&elem, &mask, NULL, 0, true);
}
```

11) 모달리스 창 만들기

현재는 모달리스(팔레트) 창 자체적으로 애드온 언로드 하는 방법을 찾지 못함
애드온 메뉴를 통해 간접적으로 애드온 언로드하는 방법을 적용한 방법을 다음에 소개함
다음 예제는 "애드온 사용법 보기" 모달리스 창에 대한 예제이다.

```
// Information.cpp

short	modelessDialogID;

// 애드온 사용법 보기
GSErrCode	showHelp (void)
{
	GSErrCode	err = NoError;

	modelessDialogID = 0;

	// 팔레트 창 열기 (모달리스 창과 호환됨)
	if ((modelessDialogID == 0) || !DGIsDialogOpen (modelessDialogID)) {
		modelessDialogID = DGModelessInit (ACAPI_GetOwnResModule (),
32501, ACAPI_GetOwnResModule (), helpHandler, 0, 1);
	}

	return err;
}

// [도움말] 애드온 사용법 보기
short DGCALLBACK helpHandler (short message, short dialogID, short item, DGUserData
/* userData */, DGMessageData /* msgData */)
{
	short	result;
	short	itmIdx;

	switch (message) {
		case DG_MSG_INIT:
			// 모달리스 창 등록
			if (ACAPI_RegisterModelessWindow (dialogID,
APIHelpPaletteAPIControlCallBack,
					API_PalEnabled_FloorPlan +
API_PalEnabled_Section + API_PalEnabled_Elevation +
					API_PalEnabled_InteriorElevation +
API_PalEnabled_Detail + API_PalEnabled_Worksheet + API_PalEnabled_3D +
API_PalEnabled_Layout) != NoError)
				DBPrintf ("Test:: ACAPI_RegisterModelessWindow failed\n");

	// ...

		case DG_MSG_CLICK:
			switch (item) {
```

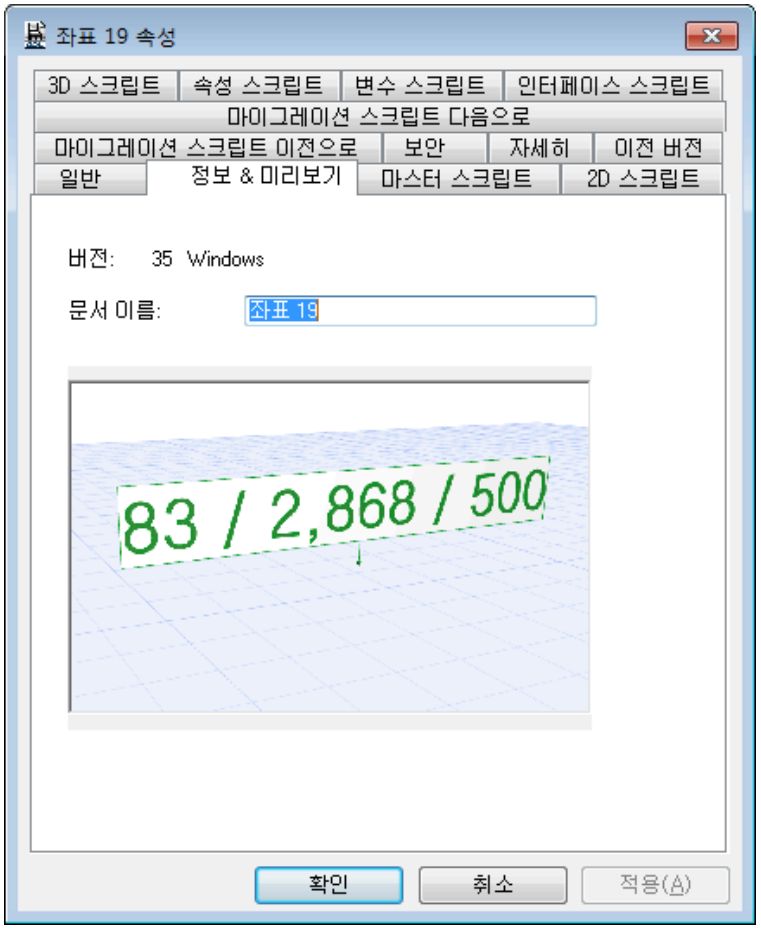
```
                                case DG_OK:
                                        ACAPI_UnregisterModelessWindow
(modelessDialogID);
                                        modelessDialogID = 0;
                                        break;

        // ...
                        }

                case DG_MSG_CLOSE:
                        ACAPI_UnregisterModelessWindow (modelessDialogID);
                        modelessDialogID = 0;
                        break;
        }

        result = item;

        return  result;
}

// 모달리스 창을 제어하기 위한 콜백함수
static GSErrCode __ACENV_CALL  APIHelpPaletteAPIControlCallBack (Int32
referenceID, API_PaletteMessageID messageID, GS::IntPtr /*param*/)
{
        if (referenceID == modelessDialogID) {
                switch (messageID) {
                        case APIPalMsg_ClosePalette:            DGModelessClose
(modelessDialogID);

break;

                        case APIPalMsg_HidePalette_Begin:       break;
                        case APIPalMsg_HidePalette_End:         break;

                        case APIPalMsg_DisableItems_Begin:      break;
                        case APIPalMsg_DisableItems_End:break;
                        case APIPalMsg_OpenPalette:                     break;
                        default:                                                break;
                }
        }

        return NoError;
}
```

```
// Main.cpp

GSErrCode __ACENV_CALLMenuCommandHandler (const API_MenuParams
*menuParams)
{
        GSErrCode     err = NoError;
```

```
	API_MenuItemRef	itemRef;
	GSFlags			itemFlags;

	switch (menuParams->menuItemRef.menuResID) {

		// ...

		case 32003:
			// 정보
			switch (menuParams->menuItemRef.itemIndex) {
				case 1:		// 애드온 사용법 보기
					BNZeroMemory (&itemRef, sizeof
(API_MenuItemRef));
					itemRef.menuResID = 32003;
					itemRef.itemIndex = 1;
					itemFlags = 0;
					ACAPI_Interface (APIIo_GetMenuItemFlagsID,
&itemRef, &itemFlags);

					extern short modelessDialogID;

					if (modelessDialogID == 0) {
						err = showHelp ();
						itemFlags |= API_MenuItemChecked;
					} else {
						if ((modelessDialogID != 0) ||
DGIsDialogOpen (modelessDialogID)) {
							DGModelessClose
(modelessDialogID);
							modelessDialogID = 0;
							itemFlags &=
~API_MenuItemChecked;
						}
					}
					ACAPI_Interface (APIIo_SetMenuItemFlagsID,
&itemRef, &itemFlags);
					return err;

					break;

			// ...

			}
			break;
	}

	return err;
}		// CommandHandler ()
```

3. 문제 해결

1) GDL 파일의 버전 정보 확인



아키캐드 버전에 따른 GDL의 버전 번호는 다음과 같습니다.

ArchiCAD 15 -- GDL v31
ArchiCAD 16 -- GDL v32
ArchiCAD 17 -- GDL v33
ArchiCAD 18 -- GDL v34
ArchiCAD 19 -- GDL v35
ArchiCAD 20 -- GDL v36
ArchiCAD 21 -- GDL v37
ArchiCAD 22 -- GDL v38

2) 리소스 작성시 유의사항

- 리소스가 변경되면, "솔루션 다시 빌드"를 하셔야 합니다.
    - 혹은 리소스 항목을 개별적으로 컴파일 한 후, 다른 소스 파일 하나만 변경한 후 "솔루션 빌드"를 하셔도 됩니다.
- 리소스 오류는 컴파일 오류에 감지되지 않습니다. 리소스에 문제가 생기면 애드온이 메뉴로 등록되지 않습니다.
- 리소스의 문자열에 비-영어 문자를 많이 사용하면 잘 안되는 경우가 많습니다. 일단 영어로 입력하고 프로그램 코드에서 한글로 후-설정하시면 됩니다.
    - 메뉴 문자열의 경우, 기존 메뉴 항목과 중복되는 경우 안 될 수 있습니다.
    - 다이얼로그의 항목의 경우, 영어로 입력하고 추후에 항목 이름을 변경하면 됩니다.

3) API 함수 작성시 유의사항

- 요소(Element) 추가, 수정, 삭제 등 데이터베이스에 변경이 발생할 경우 반드시 Undoable 세션을 열고 닫아야 합니다.
    - API v19부터는 CallUndoableCommand로 통합되었습니다.
    - 예시는 다음과 같습니다.

```
err = ACAPI_CallUndoableCommand ("Create object", [&] () -> GSErrCode {
});
```

- ACAPI_Element_Delete 함수 사용 예제

```
void deleteAllElements() {
	GSErrCode err;
	GS::Array<API_Guid> elemList;

	err = ACAPI_Element_GetElemList(API_ZombieElemID, &elemList);
	if (hasError(err)) {
		quit();
		return;
	}
	if (elemList.GetSize() > 0) {
		int size = elemList.GetSize();
		API_Elem_Head* headList = new API_Elem_Head[size];
		API_Elem_Head* toDelete;
		for (int i = 0; i < size; i++){
			API_Element el;
			el.header.guid = elemList.Get(i);
			headList[i] = el.header;
			/*
			err = ACAPI_Element_Get(&el);
			toDelete = &el.header;
			ACAPI_Element_Delete(&toDelete, 1);
			if (hasError(err)){
				quit();
				return;
```

```
                        }
                        */
                }
                ACAPI_Element_Delete(&headList, size);
                if (hasError(err)) {
                        quit();
                        return;
                }
        }
}
```

- ACAPI_Element_Change 함수 사용 예제

```
void changeAllElements() {
        GSErrCode err;
        GS::Array<API_Guid> elemList;
        API_Element elem;
        API_ElementMemo memo;
        long nElems;
        const char* foundStr;

        err = ACAPI_Element_GetElemList(API_ObjectID, &elemList);

        // 기존 배치된 Object 타입 객체의 파라미터를 변경함
        for (long xx = 0 ; xx < nElems ; ++xx) {
                BNZeroMemory (&elem, sizeof (API_Element));
                BNZeroMemory (&memo, sizeof (API_ElementMemo));
                elem.header.guid = elemList [xx];
                err = ACAPI_Element_Get(&memo);

                if (err == NoError && elem.header.hasMemo) {
                        err = ACAPI_Element_GetMemo(elem.header.guid, &memo);
                        if (err == NoError) {
                                setParameterByName (&memo, "gs_resol",
gs_resol);
                                ACAPI_Element_Change (&elem, NULL, &memo,
APIMemoMask_AddPars, true);
                        }
                        ACAPI_DisposeElemMemoHdls (&memo);
                }
        }
}
```

4) API 개발자를 위한 Helper 함수

다음은 APICommon.h, APICommon.c 파일에 내장된 함수들입니다. (일부)

```
#define     PI          3.141592653589793       // π(파이) 값
#define     EPS         1e-5                    // 실수형 비교를 위한 엡실론
```

```
void CCALL    WriteReport (const char* format, ...);  // 양식화된 정보를 창으로 보여줌
void CCALL    WriteReport_Alert (const char* format, ...);

const char*             ErrID_To_Name  (GSErrCode err);
const char*             LibID_To_Name  (API_LibTypeID typeID);
const char*             AttrID_To_Name (API_AttrTypeID typeID);
const char*             ElemID_To_Name (API_ElemTypeID typeID);
```

5) 유용한 함수

```
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdarg.h>

아래의 함수들을 사용하기 전에 필요할 경우 위의 전처리기를 추가할 것을 권장합니다.
```

```
// degree 각도를 radian 각도로 변환
double DegreeToRad (double degree)
{
        return degree * M_PI / 180;
}

// radian 각도를 degree 각도로 변환
double RadToDegree (double rad)
{
        return rad * 180 / M_PI;
}

ArchiCAD API에서 모든 요소들은 내부적으로 radian 각도를 사용합니다. 그러므로
사람에게 익숙한 degree 각도를 사용할 때에는 저장할 때 radian 각도로 변환해야 하고,
저장된 각도를 볼 때에는 degree 각도로 변환해서 표기해야 합니다. 만약 각도를
변환하지 않는다면 요소가 원하는 각도로 회전하지 않을 것입니다.
```

```
// 2차원에서 2점 간의 거리를 알려줌
double GetDistance (const API_Coord begPoint, API_Coord endPoint)
{
        return sqrt ( (begPoint.x - endPoint.x) * (begPoint.x - endPoint.x) + (begPoint.y -
endPoint.y) * (begPoint.y - endPoint.y) );
}

// 3차원에서 2점 간의 거리를 알려줌
double GetDistance (const API_Coord3D begPoint, API_Coord3D endPoint)
{
        return sqrt ( (begPoint.x - endPoint.x) * (begPoint.x - endPoint.x) + (begPoint.y -
endPoint.y) * (begPoint.y - endPoint.y) + (begPoint.z - endPoint.z) * (begPoint.z -
endPoint.z) );
}
```

```
// 선분 AB와 점 P와의 거리를 구하는 함수
double distOfPointBetweenLine (API_Coord p, API_Coord a, API_Coord b)
{
	double area;
	double dist_ab;

	area = abs ( (a.x - p.x) * (b.y - p.y) - (a.y - p.y) * (b.x - p.x) );
	dist_ab = sqrt ( (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y) );

	return  area / dist_ab;
}
```

```
// 문자열 비교
int	my_strcmp (const char *str1, const char *str2)
{
	for (; *str1 && (*str1 == *str2); ++str1, ++str2);
		return *str1 - *str2;
}
```

C언어 strcmp 함수를 사용하지 못할 경우 이 함수를 사용하시면 됩니다.
str1 문자열이 str2 문자열보다 작으면 음수, 같으면 0, 크면 양수가 나옴

```
// (p1, p2)를 이은 직선과 (p3, p4)를 이은 직선의 교차점을 구하는 함수
// Function to get intersection point with line connecting points (p1, p2) and another line (p3, p4).
API_Coord	IntersectionPoint1 (const API_Coord* p1, const API_Coord* p2, const API_Coord* p3, const API_Coord* p4)
{
	API_Coord ret;
	ret.x = ((p1->x*p2->y - p1->y*p2->x)*(p3->x - p4->x) - (p1->x - p2->x)*(p3->x*p4->y - p3->y*p4->x))/( (p1->x - p2->x)*(p3->y - p4->y) - (p1->y - p2->y)*(p3->x - p4->x) );
	ret.y = ((p1->x*p2->y - p1->y*p2->x)*(p3->y - p4->y) - (p1->y - p2->y)*(p3->x*p4->y - p3->y*p4->x)) / ( (p1->x - p2->x)*(p3->y - p4->y) - (p1->y - p2->y)*(p3->x - p4->x) );
	return  ret;
}

// y = m1*x + b1, y = m2*x + b2 두 직선의 교차점을 구하는 함수
// Function to get intersection point of two lines y = m1*x + b1, y = m2*x + b2
API_Coord	IntersectionPoint2 (double m1, double b1, double m2, double b2)
{
	API_Coord ret;
	ret.x = (b2 - b1) / (m1 - m2);
	ret.y = m1*(b2 - b1)/(m1 - m2) + b1;
	return  ret;
}

// a1*x + b1*y + c1 = 0, a2*x + b2*y + c2 = 0 두 직선의 교차점을 구하는 함수
// Function to get intersection point of two lines a1*x + b1*y + c1 = 0, a2*x + b2*y + c2 = 0
API_Coord	IntersectionPoint3 (double a1, double b1, double c1, double a2, double b2, double c2)
```

```
{
	API_Coord ret;
	ret.x = (b1*c2 - b2*c1)/(a1*b2 - a2*b1);
	ret.y = -a1/b1*(b1*c2-b2*c1)/(a1*b2-a2*b1)-c1/b1;
	return  ret;
}
```

```
// 반올림
double round (double x, int digit)
{
	return  floor((x) * pow (float (10), digit) + 0.5f) / pow (float (10), digit);
}
```

C언어 round 함수를 사용하지 못할 경우 이 함수를 사용하시면 됩니다.
예) round (1.234 * 1000 / 1000.0, 1) 의 결과는 1.200

```
// std::string 변수 값에 formatted string을 입력 받음
std::string	format_string (const std::string fmt, ...)
{
	int			size = ((int)fmt.size ()) * 2;
	va_list		ap;
	std::string	buffer;

	while (1) {
		buffer.resize(size);
		va_start (ap, fmt);
		int n = vsnprintf ((char*)buffer.data (), size, fmt.c_str (), ap);
		va_end (ap);

		if (n > -1 && n < size) {
			buffer.resize (n);
			return buffer;
		}

		if (n > -1)
			size = n + 1;
		else
			size *= 2;
	}

	return buffer;
}
```

이것은 C언어의 printf 처럼 원하는 format을 가공해서 문자열(string 클래스) 형태로 내보낼 수 있습니다.
예) format_string ("%f", 0.100);

```
// 문자열 s가 숫자로 된 문자열인지 알려줌 (숫자는 1, 문자열은 0)
short	isStringDouble (char *str)
{
	size_t size = strlen (str);
```

```
	if (size == 0)
		return 0;			// 0바이트 문자열은 숫자가 아님

	for (size_t i = 0 ; i < size ; i++) {
		if (str [i] == '.' || str [i] == '-' || str [i] == '+')
			continue;		// .-+ 문자는 무시함
		if (str [i] < '0' || str [i] > '9')
			return 0;		// 알파벳 등이 있으면 숫자 아님
	}

	return 1;	// 그밖의 경우는 숫자
}
```

```
// pName 파라미터의 값을 value로 설정함 (실수형) - 성공하면 true, 실패하면 false
bool	setParameterByName (API_ElementMemo* memo, char* pName, double value)
{
	const char*	retStr = NULL;
	int totalParams = BMGetHandleSize ((GSConstHandle)memo->params) / sizeof
(API_AddParType);		// 매개변수 수 = 핸들 크기 / 단일 핸들 크기

	for (int xx = 0 ; xx < totalParams ; ++xx) {
		retStr = GS::UniString (memo->params [0][xx].name).ToCStr ().Get ();
		if (retStr != NULL) {
			if (GS::ucscmp (GS::UniString (memo->params
[0][xx].name).ToUStr ().Get (), GS::UniString (pName).ToUStr ().Get ()) == 0) {
				memo->params [0][xx].value.real = value;
				return  true;
			}
		} else {
			return false;
		}
	}

	return  false;
}

// pName 파라미터의 값을 value로 설정함 (문자열) - 성공하면 true, 실패하면 false
bool	setParameterByName (API_ElementMemo* memo, char* pName, char* value)
{
	const char*	retStr = NULL;
	int totalParams = BMGetHandleSize ((GSConstHandle)memo->params) /
sizeof(API_AddParType);		// 매개변수 수 = 핸들 크기 / 단일 핸들 크기

	for (int xx = 0 ; xx < totalParams ; ++xx) {
		retStr = GS::UniString (memo->params [0][xx].name).ToCStr ().Get ();
		if (retStr != NULL) {
			if (GS::ucscmp (GS::UniString (memo->params
[0][xx].name).ToUStr ().Get (), GS::UniString (pName).ToUStr ().Get ()) == 0) {
				GS::ucscpy (memo->params [0][xx].value.uStr,
```

```
GS::UniString (value).ToUStr ().Get ());
                                        return  true;
                                }
                        } else {
                                return false;
                        }
                }

                return  false;
}

// pName 파라미터의 값을 가져옴 (실수형)
double getParameterValueByName (API_ElementMemo* memo, char* pName)
{
        const char*     retStr = NULL;
        int totalParams = BMGetHandleSize ((GSConstHandle)memo->params) /
sizeof(API_AddParType);                 // 매개변수 수 = 핸들 크기 / 단일 핸들 크기

        for (int xx = 0 ; xx < totalParams ; ++xx) {
                retStr = GS::UniString (memo->params [0][xx].name).ToCStr ().Get ();
                if (GS::ucscmp (GS::UniString (memo->params [0][xx].name).ToUStr ().Get
(), GS::UniString (pName).ToUStr ().Get ()) == 0) {
                        return memo->params [0][xx].value.real;
                }
        }

        return 0.0;
}

// pName 파라미터의 값을 가져옴 (문자열)
const char*     getParameterStringByName (API_ElementMemo* memo, char* pName)
{
        const char*     retStr = NULL;
        int totalParams = BMGetHandleSize ((GSConstHandle)memo->params) /
sizeof(API_AddParType);                 // 매개변수 수 = 핸들 크기 / 단일 핸들 크기

        for (int xx = 0 ; xx < totalParams ; ++xx) {
                retStr = GS::UniString (memo->params [0][xx].name).ToCStr ().Get ();
                if (GS::ucscmp (GS::UniString (memo->params [0][xx].name).ToUStr ().Get
(), GS::UniString (pName).ToUStr ().Get ()) == 0) {
                        retStr = GS::UniString (memo->params [0][xx].value.uStr).ToCStr
().Get ();
                        return retStr;
                }
        }

        retStr = "";
        return  retStr;
}
```

```
// pName 파라미터의 타입을 가져옴 - API_AddParID
API_AddParID		getParameterTypeByName (API_ElementMemo* memo, char*
pName)
{
	const char* retStr = NULL;
	int totalParams = BMGetHandleSize ((GSConstHandle)memo->params) /
sizeof(API_AddParType);		// 매개변수 수 = 핸들 크기 / 단일 핸들 크기

	for (int xx = 0 ; xx < totalParams ; ++xx) {
		retStr = GS::UniString (memo->params [0][xx].name).ToCStr ().Get ();
		if (GS::ucscmp (GS::UniString (memo->params [0][xx].name).ToUStr ().Get
(), GS::UniString (pName).ToUStr ().Get ()) == 0) {
			return memo->params [0][xx].typeID;
		}
	}

	return API_ZombieParT;
}
```

요소를 생성할 때 GDL 객체의 파라미터를 세트하거나, 요소를 읽어올 때 GDL 객체의 파라미터를 가져오는 데 사용하는 함수입니다.

```
// X, Y, Z축 방향을 선택하고, 해당 방향으로 거리를 이동한 좌표를 리턴함 (각도 고려,
단위: radian)
void		moveIn3D (char direction, double ang, double offset, double* curX, double*
curY, double* curZ)
{
	if (direction == 'x' || direction == 'X') {
		*curX = *curX + (offset * cos(ang));
		*curY = *curY + (offset * sin(ang));
	}

	if (direction == 'y' || direction == 'Y') {
		*curX = *curX - (offset * sin(ang));
		*curY = *curY + (offset * cos(ang));
	}

	if (direction == 'z' || direction == 'Z') {
		*curZ = *curZ + offset;
	}
}

// X, Y축 방향을 선택하고, 해당 방향으로 거리를 이동한 좌표를 리턴함 (각도 고려, 단위:
radian)
void		moveIn2D (char direction, double ang, double offset, double* curX, double*
curY)
{
	if (direction == 'x' || direction == 'X') {
		*curX = *curX + (offset * cos(ang));
		*curY = *curY + (offset * sin(ang));
```

```
	}

	if (direction == 'y' || direction == 'Y') {
		*curX = *curX - (offset * sin(ang));
		*curY = *curY + (offset * cos(ang));
	}
}
```

요소의 배치 위치를 변경할 때, 요소의 radian 각도와 현재 좌표를 입력하고 이동하고자 하는 축(x, y, 또는 z)과 이동 거리 offset을 입력하여 현재 좌표를 변경시킬 수 있습니다.

```
// 레이어 이름으로 레이어 인덱스 찾기
short findLayerIndex (const char* layerName)
{
	short	nLayers;
	short	xx;
	GSErrCode	err;

	API_Attribute	attrib;

	// 프로젝트 내 레이어 개수를 알아냄
	BNZeroMemory (&attrib, sizeof (API_Attribute));
	attrib.header.typeID = API_LayerID;
	err = ACAPI_Attribute_GetNum (API_LayerID, &nLayers);

	// 입력한 레이어 이름과 일치하는 레이어의 인덱스 번호 리턴
	for (xx = 1; xx <= nLayers ; ++xx) {
		attrib.header.index = xx;
		err = ACAPI_Attribute_Get (&attrib);

		if (err == NoError) {
			if (my_strcmp (attrib.layer.head.name, layerName) == 0) {
				return	attrib.layer.head.index;
			}
		}
	}

	return 0;
}
```

아키캐드 레이어 이름을 이용하여 아키캐드 내부적으로 사용되는 레이어 인덱스를 알아낼 수 있습니다.

```
// 다음은 요소(GDL 객체)를 쉽게 배치하기 위해 만든 EasyObjectPlacement 클래스임

// 클래스: 쉬운 객체 배치
class EasyObjectPlacement
{
private:
	const char*		paramName [50];
	API_AddParIDparamType [50];
```

```
        const char*             paramVal [50];
        short                   nParams;

        short                   layerInd;
        short                   floorInd;
        const GS::uchar_t*      gsmName;
public:
        double                  posX;
        double                  posY;
        double                  posZ;
        double                  radAng;

        EasyObjectPlacement ();                 // 생성자
        EasyObjectPlacement (const GS::uchar_t* gsmName, short layerInd, short
floorInd, double posX, double posY, double posZ, double radAng); // 생성자
        void            init (const GS::uchar_t* gsmName, short layerInd, short floorInd,
double posX, double posY, double posZ, double radAng);   // 초기화 함수
        void            setGsmName (const GS::uchar_t* gsmName);        // GSM 파일명
지정
        void            setLayerInd (short layerInd);  // 레이어 인덱스 지정 (1부터 시작)
        void            setFloorInd (short floorInd);   // 층 인덱스 지정 (음수, 0, 양수 가능)
        void            setParameters (short nParams, const char* paramNameList [],
API_AddParID* paramTypeList, const char* paramValList []);        // 파라미터 지정
        API_Guid        placeObject (double posX, double posY, double posZ, double
radAng);                // 객체 배치
        API_Guid        placeObject (const GS::uchar_t* gsmName, short nParams, const
char* paramNameList [], API_AddParID* paramTypeList, const char* paramValList [], short
layerInd, short floorInd, double posX, double posY, double posZ, double radAng); // 객체
배치
        API_Guid        placeObject (short nParams, ...);       // 객체 배치 (파라미터의 개수
및 파라미터 이름/타입/값만 입력)
        API_Guid        placeObject (const GS::uchar_t* gsmName, short layerInd, short
floorInd, double posX, double posY, double posZ, double radAng, short nParams, ...);
                        // 초기화를 하는 동시에 객체 배치 (가변 파라미터: 파라미터의
개수 및 파라미터 이름/타입/값만 입력)
        API_Guid        placeObjectMirrored (double posX, double posY, double posZ,
double radAng);                 // 객체 배치
        API_Guid        placeObjectMirrored (const GS::uchar_t* gsmName, short nParams,
const char* paramNameList [], API_AddParID* paramTypeList, const char* paramValList
[], short layerInd, short floorInd, double posX, double posY, double posZ, double radAng);
        // 객체 배치
        API_Guid        placeObjectMirrored (short nParams, ...);     // 객체 배치
(파라미터의 개수 및 파라미터 이름/타입/값만 입력)
        API_Guid        placeObjectMirrored (const GS::uchar_t* gsmName, short layerInd,
short floorInd, double posX, double posY, double posZ, double radAng, short nParams,
...);           // 초기화를 하는 동시에 객체 배치 (가변 파라미터: 파라미터의 개수 및
파라미터 이름/타입/값만 입력)

        // 사용법 안내
        /*
```

```
	EasyObjectPlacement objP;
	objP.init (L("유로폼v2.0.gsm"), layerInd_Euroform, infoWall.floorInd,
cell.leftBottomX, cell.leftBottomY, cell.leftBottomZ, cell.ang);

	for (xx = 0 ; xx < placementInfo.nHorEuroform ; ++xx) {
		height = 0.0;
		for (yy = 0 ; yy < placementInfo.nVerEuroform ; ++yy) {
			height += placementInfo.height [yy];
			elemList.Push (objP.placeObject (5,
				"eu_stan_onoff", APIParT_Boolean, "1.0",
				"eu_wid", APIParT_CString, format_string ("%.0f", round
(placementInfo.width [xx]*1000, 0)).c_str (),
				"eu_hei", APIParT_CString, format_string ("%.0f", round
(placementInfo.height [yy]*1000, 0)).c_str (),
				"u_ins", APIParT_CString, "벽세우기",
				"ang_x", APIParT_Angle, format_string ("%f", DegreeToRad
(90.0)).c_str ()));
			moveIn3D ('z', objP.radAng, placementInfo.height [yy], &objP.posX,
&objP.posY, &objP.posZ);
		}
		moveIn3D ('x', objP.radAng, placementInfo.width [xx], &objP.posX,
&objP.posY, &objP.posZ);
		moveIn3D ('z', objP.radAng, -height, &objP.posX, &objP.posY, &objP.posZ);
	}
	*/
};

// 클래스: 쉬운 객체 배치 - 생성자
EasyObjectPlacement::EasyObjectPlacement ()
{
}

// 클래스: 쉬운 객체 배치 - 생성자
EasyObjectPlacement::EasyObjectPlacement (const GS::uchar_t* gsmName, short
layerInd, short floorInd, double posX = 0.0, double posY = 0.0, double posZ = 0.0, double
radAng = 0.0)
{
	this->gsmName = gsmName;
	this->layerInd = layerInd;
	this->floorInd = floorInd;

	this->posX = posX;
	this->posY = posY;
	this->posZ = posZ;
	this->radAng = radAng;
}

// 클래스: 쉬운 객체 배치 - 초기화 함수
void		EasyObjectPlacement::init (const GS::uchar_t* gsmName, short layerInd,
short floorInd, double posX, double posY, double posZ, double radAng)
```

```
{
	this->gsmName = gsmName;
	this->layerInd = layerInd;
	this->floorInd = floorInd;
	this->posX = posX;
	this->posY = posY;
	this->posZ = posZ;
	this->radAng = radAng;
}

// 클래스: 쉬운 객체 배치 - GSM 파일명 지정
void		EasyObjectPlacement::setGsmName (const GS::uchar_t* gsmName)
{
	this->gsmName = gsmName;
}

// 클래스: 쉬운 객체 배치 - 레이어 인덱스 지정 (1부터 시작)
void		EasyObjectPlacement::setLayerInd (short layerInd)
{
	this->layerInd = layerInd;
}

// 클래스: 쉬운 객체 배치 - 층 인덱스 지정 (음수, 0, 양수 가능)
void		EasyObjectPlacement::setFloorInd (short floorInd)
{
	this->floorInd = floorInd;
}

// 클래스: 쉬운 객체 배치 - 파라미터 지정
void		EasyObjectPlacement::setParameters (short nParams, const char*
paramNameList [], API_AddParID* paramTypeList, const char* paramValList [])
{
	// 파라미터 개수 저장
	this->nParams = nParams;

	for (short xx = 0 ; xx < nParams ; ++xx) {
		this->paramName [xx] = paramNameList [xx];
		this->paramType [xx] = paramTypeList [xx];
		this->paramVal [xx] = paramValList [xx];
	}
}

// 클래스: 쉬운 객체 배치 - 객체 배치
API_Guid	EasyObjectPlacement::placeObject (double posX, double posY, double
posZ, double radAng)
{
	GSErrCode	err = NoError;
	API_Element			elem;
	API_ElementMemo		memo;
	API_LibPart			libPart;
```

```
	double			aParam;
	double			bParam;
	Int32			addParNum;

	char			paramName [128];
	char			paramValStr [128];
	double			paramValDouble;

	// 객체 로드
	BNZeroMemory (&elem, sizeof (API_Element));
	BNZeroMemory (&memo, sizeof (API_ElementMemo));
	BNZeroMemory (&libPart, sizeof (libPart));
	GS::ucscpy (libPart.file_UName, this->gsmName);
	err = ACAPI_LibPart_Search (&libPart, false);
	if (err != NoError)
		return elem.header.guid;
	if (libPart.location != NULL)
		delete libPart.location;

	ACAPI_LibPart_Get (&libPart);

	elem.header.typeID = API_ObjectID;
	elem.header.guid = GSGuid2APIGuid (GS::Guid (libPart.ownUnID));

	ACAPI_Element_GetDefaults (&elem, &memo);
	ACAPI_LibPart_GetParams (libPart.index, &aParam, &bParam, &addParNum,
&memo.params);

	this->posX = posX;
	this->posY = posY;
	this->posZ = posZ;
	this->radAng = radAng;

	// 라이브러리의 파라미터 값 입력
	elem.object.libInd = libPart.index;
	elem.object.reflected = false;
	elem.object.xRatio = aParam;
	elem.object.yRatio = bParam;
	elem.object.pos.x = posX;
	elem.object.pos.y = posY;
	elem.object.level = posZ;
	elem.object.angle = radAng;
	elem.header.floorInd = this->floorInd;// 층 인덱스
	elem.header.layer = this->layerInd;		// 레이어 인덱스

	for (short xx = 0 ; xx < this->nParams ; ++xx) {

		if ((this->paramType [xx] != APIParT_Separator) || (this->paramType [xx] !=
APIParT_Title) || (this->paramType [xx] != API_ZombieParT)) {
```

```
                        if (this->paramType [xx] == APIParT_CString) {
                                sprintf (paramName, "%s", this->paramName [xx]);
                                sprintf (paramValStr, "%s", this->paramVal [xx]);
                                setParameterByName (&memo, paramName, paramValStr);
                        } else {
                                sprintf (paramName, "%s", this->paramName [xx]);
                                paramValDouble = atof (this->paramVal [xx]);
                                setParameterByName (&memo, paramName,
paramValDouble);
                                if (my_strcmp (paramName, "A") == 0)
elem.object.xRatio = paramValDouble;
                                if (my_strcmp (paramName, "B") == 0)
elem.object.yRatio = paramValDouble;
                        }
                }
        }

        // 객체 배치
        ACAPI_Element_Create (&elem, &memo);
        ACAPI_DisposeElemMemoHdls (&memo);

        return  elem.header.guid;
}

// 클래스: 쉬운 객체 배치 - 객체 배치
API_Guid        EasyObjectPlacement::placeObject (const GS::uchar_t* gsmName, short
nParams, const char* paramNameList [], API_AddParID* paramTypeList, const char*
paramValList [], short layerInd, short floorInd, double posX, double posY, double posZ,
double radAng)
{
        GSErrCode     err = NoError;
        API_Element                     elem;
        API_ElementMemo                 memo;
        API_LibPart                     libPart;

        double                          aParam;
        double                          bParam;
        Int32                           addParNum;

        char                            paramName [128];
        char                            paramValStr [128];
        double                          paramValDouble;

        // 파라미터 개수 저장
        this->nParams = nParams;

        // 객체 로드
        BNZeroMemory (&elem, sizeof (API_Element));
        BNZeroMemory (&memo, sizeof (API_ElementMemo));
        BNZeroMemory (&libPart, sizeof (libPart));
```

```
	this->gsmName = gsmName;
	GS::ucscpy (libPart.file_UName, gsmName);
	err = ACAPI_LibPart_Search (&libPart, false);
	if (err != NoError)
		return elem.header.guid;
	if (libPart.location != NULL)
		delete libPart.location;

	ACAPI_LibPart_Get (&libPart);

	elem.header.typeID = API_ObjectID;
	elem.header.guid = GSGuid2APIGuid (GS::Guid (libPart.ownUnID));

	ACAPI_Element_GetDefaults (&elem, &memo);
	ACAPI_LibPart_GetParams (libPart.index, &aParam, &bParam, &addParNum,
&memo.params);

	this->posX = posX;
	this->posY = posY;
	this->posZ = posZ;
	this->radAng = radAng;

	// 라이브러리의 파라미터 값 입력
	elem.object.libInd = libPart.index;
	elem.object.reflected = false;
	elem.object.xRatio = aParam;
	elem.object.yRatio = bParam;
	elem.object.pos.x = posX;
	elem.object.pos.y = posY;
	elem.object.level = posZ;
	elem.object.angle = radAng;
	this->floorInd = elem.header.floorInd = floorInd;	// 층 인덱스
	this->layerInd = elem.header.layer = layerInd;		// 레이어 인덱스

	for (short xx = 0 ; xx < nParams ; ++xx) {

		this->paramName [xx] = paramNameList [xx];
		this->paramType [xx] = paramTypeList [xx];
		this->paramVal [xx] = paramValList [xx];

		if ((paramTypeList [xx] != APIParT_Separator) || (paramTypeList [xx] !=
APIParT_Title) || (paramTypeList [xx] != API_ZombieParT)) {
			if (paramTypeList [xx] == APIParT_CString) {
				sprintf (paramName, "%s", paramNameList [xx]);
				sprintf (paramValStr, "%s", paramValList [xx]);
				setParameterByName (&memo, paramName, paramValStr);
			} else {
				sprintf (paramName, "%s", paramNameList [xx]);
				paramValDouble = atof (paramValList [xx]);
				setParameterByName (&memo, paramName,
```

```
paramValDouble);
                                if (my_strcmp (paramName, "A") == 0)
elem.object.xRatio = paramValDouble;
                                if (my_strcmp (paramName, "B") == 0)
elem.object.yRatio = paramValDouble;
                            }
                    }
            }

            // 객체 배치
            ACAPI_Element_Create (&elem, &memo);
            ACAPI_DisposeElemMemoHdls (&memo);

            return  elem.header.guid;
}

// 클래스: 쉬운 객체 배치 - 객체 배치 (파라미터의 개수 및 파라미터 이름/타입/값만 입력)
API_Guid        EasyObjectPlacement::placeObject (short nParams, ...)
{
            GSErrCode     err = NoError;
            API_Element                 elem;
            API_ElementMemo             memo;
            API_LibPart                 libPart;

            double                      aParam;
            double                      bParam;
            Int32                       addParNum;

            char                        paramName [128];
            char                        paramValStr [128];
            double                      paramValDouble;

            // 파라미터 개수 저장
            this->nParams = nParams;

            // 객체 로드
            BNZeroMemory (&elem, sizeof (API_Element));
            BNZeroMemory (&memo, sizeof (API_ElementMemo));
            BNZeroMemory (&libPart, sizeof (libPart));
            GS::ucscpy (libPart.file_UName, this->gsmName);
            err = ACAPI_LibPart_Search (&libPart, false);
            if (err != NoError)
                    return elem.header.guid;
            if (libPart.location != NULL)
                    delete libPart.location;

            ACAPI_LibPart_Get (&libPart);

            elem.header.typeID = API_ObjectID;
            elem.header.guid = GSGuid2APIGuid (GS::Guid (libPart.ownUnID));
```

```
	ACAPI_Element_GetDefaults (&elem, &memo);
	ACAPI_LibPart_GetParams (libPart.index, &aParam, &bParam, &addParNum,
&memo.params);

	// 라이브러리의 파라미터 값 입력
	elem.object.libInd = libPart.index;
	elem.object.reflected = false;
	elem.object.xRatio = aParam;
	elem.object.yRatio = bParam;
	elem.object.pos.x = this->posX;
	elem.object.pos.y = this->posY;
	elem.object.level = this->posZ;
	elem.object.angle = this->radAng;
	elem.header.floorInd = this->floorInd;// 층 인덱스
	elem.header.layer = this->layerInd;		// 레이어 인덱스

	va_list ap;

	va_start (ap, nParams);

	for (short xx = 0 ; xx < this->nParams ; ++xx) {

		this->paramName [xx] = va_arg (ap, char*);
		this->paramType [xx] = va_arg (ap, API_AddParID);
		this->paramVal [xx] = va_arg (ap, char*);

		if ((this->paramType [xx] != APIParT_Separator) || (this->paramType [xx] !=
APIParT_Title) || (this->paramType [xx] != API_ZombieParT)) {
			if (this->paramType [xx] == APIParT_CString) {
				sprintf (paramName, "%s", this->paramName [xx]);
				sprintf (paramValStr, "%s", this->paramVal [xx]);
				setParameterByName (&memo, paramName, paramValStr);
			} else {
				sprintf (paramName, "%s", this->paramName [xx]);
				paramValDouble = atof (this->paramVal [xx]);
				setParameterByName (&memo, paramName,
paramValDouble);
				if (my_strcmp (paramName, "A") == 0)
elem.object.xRatio = paramValDouble;
				if (my_strcmp (paramName, "B") == 0)
elem.object.yRatio = paramValDouble;
			}
		}
	}

	va_end (ap);

	// 객체 배치
	ACAPI_Element_Create (&elem, &memo);
```

```
	ACAPI_DisposeElemMemoHdls (&memo);

	return  elem.header.guid;
}

// 클래스: 쉬운 객체 배치 - 초기화를 하는 동시에 객체 배치 (가변 파라미터: 파라미터의 개수 및 파라미터 이름/타입/값만 입력)
API_Guid	EasyObjectPlacement::placeObject (const GS::uchar_t* gsmName, short layerInd, short floorInd, double posX, double posY, double posZ, double radAng, short nParams, ...)
{
	this->init (gsmName, layerInd, floorInd, posX, posY, posZ, radAng);

	GSErrCode	err = NoError;
	API_Element			elem;
	API_ElementMemo		memo;
	API_LibPart			libPart;

	double				aParam;
	double				bParam;
	Int32				addParNum;

	char				paramName [128];
	char				paramValStr [128];
	double				paramValDouble;

	// 파라미터 개수 저장
	this->nParams = nParams;

	// 객체 로드
	BNZeroMemory (&elem, sizeof (API_Element));
	BNZeroMemory (&memo, sizeof (API_ElementMemo));
	BNZeroMemory (&libPart, sizeof (libPart));
	GS::ucscpy (libPart.file_UName, this->gsmName);
	err = ACAPI_LibPart_Search (&libPart, false);
	if (err != NoError)
		return elem.header.guid;
	if (libPart.location != NULL)
		delete libPart.location;

	ACAPI_LibPart_Get (&libPart);

	elem.header.typeID = API_ObjectID;
	elem.header.guid = GSGuid2APIGuid (GS::Guid (libPart.ownUnID));

	ACAPI_Element_GetDefaults (&elem, &memo);
	ACAPI_LibPart_GetParams (libPart.index, &aParam, &bParam, &addParNum, &memo.params);

	// 라이브러리의 파라미터 값 입력
```

```
	elem.object.libInd = libPart.index;
	elem.object.reflected = false;
	elem.object.xRatio = aParam;
	elem.object.yRatio = bParam;
	elem.object.pos.x = this->posX;
	elem.object.pos.y = this->posY;
	elem.object.level = this->posZ;
	elem.object.angle = this->radAng;
	elem.header.floorInd = this->floorInd;// 층 인덱스
	elem.header.layer = this->layerInd;		// 레이어 인덱스

	va_list ap;

	va_start (ap, nParams);

	for (short xx = 0 ; xx < this->nParams ; ++xx) {

		this->paramName [xx] = va_arg (ap, char*);
		this->paramType [xx] = va_arg (ap, API_AddParID);
		this->paramVal [xx] = va_arg (ap, char*);

		if ((this->paramType [xx] != APIParT_Separator) || (this->paramType [xx] !=
APIParT_Title) || (this->paramType [xx] != API_ZombieParT)) {
			if (this->paramType [xx] == APIParT_CString) {
				sprintf (paramName, "%s", this->paramName [xx]);
				sprintf (paramValStr, "%s", this->paramVal [xx]);
				setParameterByName (&memo, paramName, paramValStr);
			} else {
				sprintf (paramName, "%s", this->paramName [xx]);
				paramValDouble = atof (this->paramVal [xx]);
				setParameterByName (&memo, paramName,
paramValDouble);
				if (my_strcmp (paramName, "A") == 0)
elem.object.xRatio = paramValDouble;
				if (my_strcmp (paramName, "B") == 0)
elem.object.yRatio = paramValDouble;
			}
		}
	}

	va_end (ap);

	// 객체 배치
	ACAPI_Element_Create (&elem, &memo);
	ACAPI_DisposeElemMemoHdls (&memo);

	return  elem.header.guid;
}

// 클래스: 쉬운 객체 배치 - 객체 배치
```

```
API_Guid	EasyObjectPlacement::placeObjectMirrored (double posX, double posY, double posZ, double radAng)
{
	GSErrCode	err = NoError;
	API_Element			elem;
	API_ElementMemo		memo;
	API_LibPart			libPart;

	double				aParam;
	double				bParam;
	Int32				addParNum;

	char				paramName [128];
	char				paramValStr [128];
	double				paramValDouble;

	// 객체 로드
	BNZeroMemory (&elem, sizeof (API_Element));
	BNZeroMemory (&memo, sizeof (API_ElementMemo));
	BNZeroMemory (&libPart, sizeof (libPart));
	GS::ucscpy (libPart.file_UName, this->gsmName);
	err = ACAPI_LibPart_Search (&libPart, false);
	if (err != NoError)
		return elem.header.guid;
	if (libPart.location != NULL)
		delete libPart.location;

	ACAPI_LibPart_Get (&libPart);

	elem.header.typeID = API_ObjectID;
	elem.header.guid = GSGuid2APIGuid (GS::Guid (libPart.ownUnID));

	ACAPI_Element_GetDefaults (&elem, &memo);
	ACAPI_LibPart_GetParams (libPart.index, &aParam, &bParam, &addParNum, &memo.params);

	this->posX = posX;
	this->posY = posY;
	this->posZ = posZ;
	this->radAng = radAng;

	// 라이브러리의 파라미터 값 입력
	elem.object.libInd = libPart.index;
	elem.object.reflected = true;
	elem.object.xRatio = aParam;
	elem.object.yRatio = bParam;
	elem.object.pos.x = posX;
	elem.object.pos.y = posY;
	elem.object.level = posZ;
	elem.object.angle = radAng;
```

```
	elem.header.floorInd = this->floorInd;// 층 인덱스
	elem.header.layer = this->layerInd;		// 레이어 인덱스

	for (short xx = 0 ; xx < this->nParams ; ++xx) {

		if ((this->paramType [xx] != APIParT_Separator) || (this->paramType [xx] !=
APIParT_Title) || (this->paramType [xx] != API_ZombieParT)) {
			if (this->paramType [xx] == APIParT_CString) {
				sprintf (paramName, "%s", this->paramName [xx]);
				sprintf (paramValStr, "%s", this->paramVal [xx]);
				setParameterByName (&memo, paramName, paramValStr);
			} else {
				sprintf (paramName, "%s", this->paramName [xx]);
				paramValDouble = atof (this->paramVal [xx]);
				setParameterByName (&memo, paramName,
paramValDouble);
				if (my_strcmp (paramName, "A") == 0)
elem.object.xRatio = paramValDouble;
				if (my_strcmp (paramName, "B") == 0)
elem.object.yRatio = paramValDouble;
			}
		}
	}

	// 객체 배치
	ACAPI_Element_Create (&elem, &memo);
	ACAPI_DisposeElemMemoHdls (&memo);

	return  elem.header.guid;
}

// 클래스: 쉬운 객체 배치 - 객체 배치
API_Guid	EasyObjectPlacement::placeObjectMirrored (const GS::uchar_t*
gsmName, short nParams, const char* paramNameList [], API_AddParID* paramTypeList,
const char* paramValList [], short layerInd, short floorInd, double posX, double posY,
double posZ, double radAng)
{
	GSErrCode	err = NoError;
	API_Element			elem;
	API_ElementMemo		memo;
	API_LibPart			libPart;

	double				aParam;
	double				bParam;
	Int32				addParNum;

	char				paramName [128];
	char				paramValStr [128];
	double				paramValDouble;
```

```
	// 파라미터 개수 저장
	this->nParams = nParams;

	// 객체 로드
	BNZeroMemory (&elem, sizeof (API_Element));
	BNZeroMemory (&memo, sizeof (API_ElementMemo));
	BNZeroMemory (&libPart, sizeof (libPart));
	this->gsmName = gsmName;
	GS::ucscpy (libPart.file_UName, gsmName);
	err = ACAPI_LibPart_Search (&libPart, false);
	if (err != NoError)
		return elem.header.guid;
	if (libPart.location != NULL)
		delete libPart.location;

	ACAPI_LibPart_Get (&libPart);

	elem.header.typeID = API_ObjectID;
	elem.header.guid = GSGuid2APIGuid (GS::Guid (libPart.ownUnID));

	ACAPI_Element_GetDefaults (&elem, &memo);
	ACAPI_LibPart_GetParams (libPart.index, &aParam, &bParam, &addParNum,
&memo.params);

	this->posX = posX;
	this->posY = posY;
	this->posZ = posZ;
	this->radAng = radAng;

	// 라이브러리의 파라미터 값 입력
	elem.object.libInd = libPart.index;
	elem.object.reflected = true;
	elem.object.xRatio = aParam;
	elem.object.yRatio = bParam;
	elem.object.pos.x = posX;
	elem.object.pos.y = posY;
	elem.object.level = posZ;
	elem.object.angle = radAng;
	this->floorInd = elem.header.floorInd = floorInd;	// 층 인덱스
	this->layerInd = elem.header.layer = layerInd;		// 레이어 인덱스

	for (short xx = 0 ; xx < nParams ; ++xx) {

		this->paramName [xx] = paramNameList [xx];
		this->paramType [xx] = paramTypeList [xx];
		this->paramVal [xx] = paramValList [xx];

		if ((paramTypeList [xx] != APIParT_Separator) || (paramTypeList [xx] !=
APIParT_Title) || (paramTypeList [xx] != API_ZombieParT)) {
			if (paramTypeList [xx] == APIParT_CString) {
```

```
                                sprintf (paramName, "%s", paramNameList [xx]);
                                sprintf (paramValStr, "%s", paramValList [xx]);
                                setParameterByName (&memo, paramName, paramValStr);
                        } else {
                                sprintf (paramName, "%s", paramNameList [xx]);
                                paramValDouble = atof (paramValList [xx]);
                                setParameterByName (&memo, paramName,
paramValDouble);
                                if (my_strcmp (paramName, "A") == 0)
elem.object.xRatio = paramValDouble;
                                if (my_strcmp (paramName, "B") == 0)
elem.object.yRatio = paramValDouble;
                        }
                }
        }

        // 객체 배치
        ACAPI_Element_Create (&elem, &memo);
        ACAPI_DisposeElemMemoHdls (&memo);

        return  elem.header.guid;
}

// 클래스: 쉬운 객체 배치 - 객체 배치 (파라미터의 개수 및 파라미터 이름/타입/값만 입력)
API_Guid        EasyObjectPlacement::placeObjectMirrored (short nParams, ...)
{
        GSErrCode     err = NoError;
        API_Element                     elem;
        API_ElementMemo                 memo;
        API_LibPart                     libPart;

        double                          aParam;
        double                          bParam;
        Int32                           addParNum;

        char                            paramName [128];
        char                            paramValStr [128];
        double                          paramValDouble;

        // 파라미터 개수 저장
        this->nParams = nParams;

        // 객체 로드
        BNZeroMemory (&elem, sizeof (API_Element));
        BNZeroMemory (&memo, sizeof (API_ElementMemo));
        BNZeroMemory (&libPart, sizeof (libPart));
        GS::ucscpy (libPart.file_UName, this->gsmName);
        err = ACAPI_LibPart_Search (&libPart, false);
        if (err != NoError)
                return elem.header.guid;
```

```
	if (libPart.location != NULL)
		delete libPart.location;

	ACAPI_LibPart_Get (&libPart);

	elem.header.typeID = API_ObjectID;
	elem.header.guid = GSGuid2APIGuid (GS::Guid (libPart.ownUnID));

	ACAPI_Element_GetDefaults (&elem, &memo);
	ACAPI_LibPart_GetParams (libPart.index, &aParam, &bParam, &addParNum,
&memo.params);

	// 라이브러리의 파라미터 값 입력
	elem.object.libInd = libPart.index;
	elem.object.reflected = true;
	elem.object.xRatio = aParam;
	elem.object.yRatio = bParam;
	elem.object.pos.x = this->posX;
	elem.object.pos.y = this->posY;
	elem.object.level = this->posZ;
	elem.object.angle = this->radAng;
	elem.header.floorInd = this->floorInd;// 층 인덱스
	elem.header.layer = this->layerInd;		// 레이어 인덱스

	va_list ap;

	va_start (ap, nParams);

	for (short xx = 0 ; xx < this->nParams ; ++xx) {

		this->paramName [xx] = va_arg (ap, char*);
		this->paramType [xx] = va_arg (ap, API_AddParID);
		this->paramVal [xx] = va_arg (ap, char*);

		if ((this->paramType [xx] != APIParT_Separator) || (this->paramType [xx] !=
APIParT_Title) || (this->paramType [xx] != API_ZombieParT)) {
			if (this->paramType [xx] == APIParT_CString) {
				sprintf (paramName, "%s", this->paramName [xx]);
				sprintf (paramValStr, "%s", this->paramVal [xx]);
				setParameterByName (&memo, paramName, paramValStr);
			} else {
				sprintf (paramName, "%s", this->paramName [xx]);
				paramValDouble = atof (this->paramVal [xx]);
				setParameterByName (&memo, paramName,
paramValDouble);
				if (my_strcmp (paramName, "A") == 0)
elem.object.xRatio = paramValDouble;
				if (my_strcmp (paramName, "B") == 0)
elem.object.yRatio = paramValDouble;
			}
```

```
		}
	}

	va_end (ap);

	// 객체 배치
	ACAPI_Element_Create (&elem, &memo);
	ACAPI_DisposeElemMemoHdls (&memo);

	return  elem.header.guid;
}

// 클래스: 쉬운 객체 배치 - 초기화를 하는 동시에 객체 배치 (가변 파라미터: 파라미터의 개수 및 파라미터 이름/타입/값만 입력)
API_Guid	EasyObjectPlacement::placeObjectMirrored (const GS::uchar_t* gsmName, short layerInd, short floorInd, double posX, double posY, double posZ, double radAng, short nParams, ...)
{
	this->init (gsmName, layerInd, floorInd, posX, posY, posZ, radAng);

	GSErrCode	err = NoError;
	API_Element			elem;
	API_ElementMemo		memo;
	API_LibPart			libPart;

	double				aParam;
	double				bParam;
	Int32				addParNum;

	char				paramName [128];
	char				paramValStr [128];
	double				paramValDouble;

	// 파라미터 개수 저장
	this->nParams = nParams;

	// 객체 로드
	BNZeroMemory (&elem, sizeof (API_Element));
	BNZeroMemory (&memo, sizeof (API_ElementMemo));
	BNZeroMemory (&libPart, sizeof (libPart));
	GS::ucscpy (libPart.file_UName, this->gsmName);
	err = ACAPI_LibPart_Search (&libPart, false);
	if (err != NoError)
		return elem.header.guid;
	if (libPart.location != NULL)
		delete libPart.location;

	ACAPI_LibPart_Get (&libPart);

	elem.header.typeID = API_ObjectID;
```

```
	elem.header.guid = GSGuid2APIGuid (GS::Guid (libPart.ownUnID));

	ACAPI_Element_GetDefaults (&elem, &memo);
	ACAPI_LibPart_GetParams (libPart.index, &aParam, &bParam, &addParNum,
&memo.params);

	// 라이브러리의 파라미터 값 입력
	elem.object.libInd = libPart.index;
	elem.object.reflected = true;
	elem.object.xRatio = aParam;
	elem.object.yRatio = bParam;
	elem.object.pos.x = this->posX;
	elem.object.pos.y = this->posY;
	elem.object.level = this->posZ;
	elem.object.angle = this->radAng;
	elem.header.floorInd = this->floorInd;// 층 인덱스
	elem.header.layer = this->layerInd;		// 레이어 인덱스

	va_list ap;

	va_start (ap, nParams);

	for (short xx = 0 ; xx < this->nParams ; ++xx) {

		this->paramName [xx] = va_arg (ap, char*);
		this->paramType [xx] = va_arg (ap, API_AddParID);
		this->paramVal [xx] = va_arg (ap, char*);

		if ((this->paramType [xx] != APIParT_Separator) || (this->paramType [xx] !=
APIParT_Title) || (this->paramType [xx] != API_ZombieParT)) {
			if (this->paramType [xx] == APIParT_CString) {
				sprintf (paramName, "%s", this->paramName [xx]);
				sprintf (paramValStr, "%s", this->paramVal [xx]);
				setParameterByName (&memo, paramName, paramValStr);
			} else {
				sprintf (paramName, "%s", this->paramName [xx]);
				paramValDouble = atof (this->paramVal [xx]);
				setParameterByName (&memo, paramName,
paramValDouble);
				if (my_strcmp (paramName, "A") == 0)
elem.object.xRatio = paramValDouble;
				if (my_strcmp (paramName, "B") == 0)
elem.object.yRatio = paramValDouble;
			}
		}
	}

	va_end (ap);

	// 객체 배치
```

```
	ACAPI_Element_Create (&elem, &memo);
	ACAPI_DisposeElemMemoHdls (&memo);

	return  elem.header.guid;
}
```

```
// 모듈 데이터 저장, 로드하기
// 이것을 이용하면 특정 상태를 아키캐드 내부적으로 저장하고 로드할 수 있습니다.
// (별도의 파일을 생성하지 않음)

// 임의의 변수나 구조체를 사용하면 됨
StatusOfLayerNameSystem  selectedInfoSaved;    // 저장된 버튼 상태를 여기로 로드함

void    Load () {
	API_ModulData  info;

	BNZeroMemory (&info, sizeof (API_ModulData));
	err = ACAPI_ModulData_Get (&info, "ButtonStatus");

	if (err == NoError && info.dataVersion == 1) {
		selectedInfoSaved = *(reinterpret_cast<StatusOfLayerNameSystem*>
(*info.dataHdl));

		// 여기서 구체적인 로드 작업을 진행함
	}

	BMKillHandle (&info.dataHdl);
}

void    Save () {
	GSErrCode err = NoError;

	short   xx;
	API_ModulData        info;
	BNZeroMemory (&info, sizeof (API_ModulData));
	info.dataVersion = 1;
	info.platformSign = GS::Act_Platform_Sign;
	info.dataHdl = BMAllocateHandle (sizeof (StatusOfLayerNameSystem), 0, 0);

	if (info.dataHdl != NULL) {
		// 여기서 구체적인 저장 작업을 진행함

		*(reinterpret_cast<StatusOfLayerNameSystem*> (*info.dataHdl)) =
selectedInfoSaved;
		err = ACAPI_ModulData_Store (&info, "ButtonStatus");
		BMKillHandle (&info.dataHdl);
	} else {
		err = APIERR_MEMFULL;
	}
```

```
        return  err;
}
```

```
// 오름차순으로 정렬할 때 사용하는 비교함수 (퀵소트 qsort에 사용됨)
int compare (const void* first, const void* second)
{
    if (*(double*)first > *(double*)second)
        return 1;
    else if (*(double*)first < *(double*)second)
        return -1;
    else
        return 0;
}
```

3. 새로운 기능

1) 데이터베이스 연동 (SQLite 3)

- SQLite 다운로드 페이지에서 2가지를 다운로드한다.
    - Source Code zip 파일
    - Precompiled Binaries for Windows x64 zip 파일
- 압축 파일을 푼 다음 .c 확장자 파일만 지운다.
- Developer Command Prompt를 실행한다.
    - sqlite3 디렉토리로 이동한 후 다음 명령어를 실행한다.
    - lib /def:sqlite3.def /machine:x64
    - 결과물로 다음 파일이 나옴: sqlite3.exp, sqlite3.lib
- 파일들을 솔루션 소스 디렉토리 안에 복사한다.
    - sqlite3.def, sqlite3.dll, sqlite3.exp, sqlite3.h, sqlite3.lib, sqlite3ext.h
    - 예시) C:\Users\peace\OneDrive\문서\Visual Studio 2010\Projects\MaxBIM\MaxBIM\Src\sqlite3
- 프로젝트의 구성 속성을 변경한다.
    - C/C++ > 추가 포함 디렉터리에 다음을 추가: $(SolutionDir)MaxBIM\Src\sqlite3
    - 링커 > 추가 라이브러리 디렉터리에 다음을 추가: $(SolutionDir)MaxBIM\Src\sqlite3
    - 링커 > 입력에 다음을 추가: sqlite3.lib
- 소스 파일에 다음 구문을 추가하고 컴파일해서 성공하면 완료
    - #inlcude "sqlite3.h"
- sqlite3.dll 파일을 아키캐드 설치 디렉토리에 복사한다.
    - 예시) C:\Program Files\GRAPHISOFT\ArchiCAD 19
    - db 파일을 생성하면 위의 디렉토리에 생성될 것이다.