**Пример №9**

Задача: построение трубы (сдвиг)

**Поэтапное построение эскиза.**

1. Создаем функцию CreateSketch:

void CreateSketch(RPArray<MbContour>& \_arrContours)

{

...

}

2. Затем описываем CreateSketch (Все функции, которые мы применили объяснены в комментариях).

const double RAD\_OUTER = 35.0; // Радиус внешней окружности

const double RAD\_INNER = 30.0; // Радиус внутренней окружности

MbArc\* pCircleExt = new MbArc(MbCartPoint(0, 0), RAD\_OUTER);

MbArc\* pCircleInt = new MbArc(MbCartPoint(0, 0), RAD\_INNER);

\_arrContours.push\_back(new MbContour(\*pCircleExt, true));

\_arrContours.push\_back(new MbContour(\*pCircleInt, true));

3. Переходим к описанию основной функции.

void MakeUserCommand0()

{

...

}

4. Первое что мы делам, создаем локальную систему координат, массив контрольных точек для NURBS-сплайна .

// Локальная СК (по умолчанию совпадает с мировой СК)

MbPlacement3D pl;

// Массив контрольных точек для построения NURBS-сплайна

std::vector<MbCartPoint3D> vecPnts = { { 0, 0, 0 }, { 0, 0, 150 } };

SArray<MbCartPoint3D> arrPnts(vecPnts);

// Построение направляющей кривой в виде незамкнутого NURBS-сплайна

// 2-го порядка по контрольным точкам

MbNurbs3D\* pSpline = MbNurbs3D::Create(2, arrPnts, false);

// Описание образующей кривой в виде плоского контура на плоскости XY мировой СК

MbPlane\* pPlaneXY = new MbPlane(MbCartPoint3D(0, 0, 0), MbCartPoint3D(1, 0, 0),

MbCartPoint3D(0, 1, 0));

5. Далее, мы вызываем функцию отвечающую за создание эскиза.

// Построение образующей кривой с помощью вспомогательной функции CreateSketch

RPArray<MbContour> arrContours;

CreateSketch(arrContours);

MbSweptData sweptData(\*pPlaneXY, arrContours);

// Объект с параметрами операции построения тела заметания

EvolutionValues params;

// Вариант плоскопараллельного движения образующей вдоль направляющей

params.parallel = 0;

// Служебные объекты-именователи для вызова геометрической операции

MbSNameMaker operNames(ct\_CurveEvolutionSolid, MbSNameMaker::i\_SideNone, 0);

MbSNameMaker cNames(ct\_CurveSweptSolid, MbSNameMaker::i\_SideNone, 0);

PArray<MbSNameMaker> contourNames(1, 0, false);

contourNames.Add(&cNames);

MbSNameMaker splineNames(ct\_CurveSweptSolid, MbSNameMaker::i\_SideNone, 0);

6. Вызываем операцию для построения тела заметания.

// Вызов операции построения тела заметания

MbSolid\* pSolid = NULL;

MbResultType res = ::EvolutionSolid(sweptData, \*pSpline, params, operNames,

contourNames, splineNames, pSolid);

// Отображение построенного тела

if (res == rt\_Success)

viewManager->AddObject(Style(1, LIGHTGRAY), pSolid);

// Отображение направляющей кривой со смещением вдоль оси Y (в целях отображения,

// чтобы отображаемая кривая была смещена от поверхности тела).

pSpline->Move(MbVector3D(MbCartPoint3D(0, 0, 0), MbCartPoint3D(0, -50, 0)));

viewManager->AddObject(Style(3, RGB(0, 0, 255)), pSpline, &pl);

7. Уменьшаем счетчики ссылок.

// Уменьшение счетчиков ссылок объектов ядра

::DeleteItem(pSolid);

::DeleteItem(pPlaneXY);

::DeleteItem(pSpline);

**Код программы**

void CreateSketch(RPArray<MbContour>& \_arrContours)

{

const double RAD\_OUTER = 35.0; // Радиус внешней окружности

const double RAD\_INNER = 30.0; // Радиус внутренней окружности

MbArc\* pCircleExt = new MbArc(MbCartPoint(0, 0), RAD\_OUTER);

MbArc\* pCircleInt = new MbArc(MbCartPoint(0, 0), RAD\_INNER);

\_arrContours.push\_back(new MbContour(\*pCircleExt, true));

\_arrContours.push\_back(new MbContour(\*pCircleInt, true));

}

void MakeUserCommand0()

{

// Локальная СК (по умолчанию совпадает с мировой СК)

MbPlacement3D pl;

// Массив контрольных точек для построения NURBS-сплайна

std::vector<MbCartPoint3D> vecPnts = { { 0, 0, 0 }, { 0, 0, 150 } };

SArray<MbCartPoint3D> arrPnts(vecPnts);

// Построение направляющей кривой в виде незамкнутого NURBS-сплайна

// 2-го порядка по контрольным точкам

MbNurbs3D\* pSpline = MbNurbs3D::Create(2, arrPnts, false);

// Описание образующей кривой в виде плоского контура на плоскости XY мировой СК

MbPlane\* pPlaneXY = new MbPlane(MbCartPoint3D(0, 0, 0), MbCartPoint3D(1, 0, 0),

MbCartPoint3D(0, 1, 0));

// Построение образующей кривой с помощью вспомогательной функции CreateSketch

RPArray<MbContour> arrContours;

CreateSketch(arrContours);

MbSweptData sweptData(\*pPlaneXY, arrContours);

// Объект с параметрами операции построения тела заметания

EvolutionValues params;

// Вариант плоскопараллельного движения образующей вдоль направляющей

params.parallel = 0;

// Служебные объекты-именователи для вызова геометрической операции

MbSNameMaker operNames(ct\_CurveEvolutionSolid, MbSNameMaker::i\_SideNone, 0);

MbSNameMaker cNames(ct\_CurveSweptSolid, MbSNameMaker::i\_SideNone, 0);

PArray<MbSNameMaker> contourNames(1, 0, false);

contourNames.Add(&cNames);

MbSNameMaker splineNames(ct\_CurveSweptSolid, MbSNameMaker::i\_SideNone, 0);

// Вызов операции построения тела заметания

MbSolid\* pSolid = NULL;

MbResultType res = ::EvolutionSolid(sweptData, \*pSpline, params, operNames,

contourNames, splineNames, pSolid);

// Отображение построенного тела

if (res == rt\_Success)

viewManager->AddObject(Style(1, LIGHTGRAY), pSolid);

// Отображение направляющей кривой со смещением вдоль оси Y (в целях отображения,

// чтобы отображаемая кривая была смещена от поверхности тела).

pSpline->Move(MbVector3D(MbCartPoint3D(0, 0, 0), MbCartPoint3D(0, -50, 0)));

viewManager->AddObject(Style(3, RGB(0, 0, 255)), pSpline, &pl);

// Уменьшение счетчиков ссылок объектов ядра

::DeleteItem(pSolid);

::DeleteItem(pPlaneXY);

::DeleteItem(pSpline);

}

**Список используемых функций**

CreateSketch – пользовательская функция для создание эскиза.

MbResultType::EvolutionSolid - функция для построения тела заметания.

**Список используемых классов**

MbArc – класс для создания окружностей.

MbNurbs3D – класс для создания NURBS-сплайна

MbPlane – класс плоскости.

vector – класс для контейнеров последовательности.

MbCartPoint – класс точки в 2-ух мерном пространстве.

MbSweptData – класс данных образующей.

MbSolid – класс твердого тела.

MbSNameMaker – класс для генерирования имен.

MbResultType – класс для описания конечного результата ( в данном случае твердое тело выдавливания)