Проектная деятельность

Разработка ИЭТР на базе графического ядра C3D

Участники - студенты 2-го курса:
- Фролов А. М. группы 181-326
- Серяков А. В. группы 181-326
- Волобуев Р. А. группы 181-326

Цель проекта:

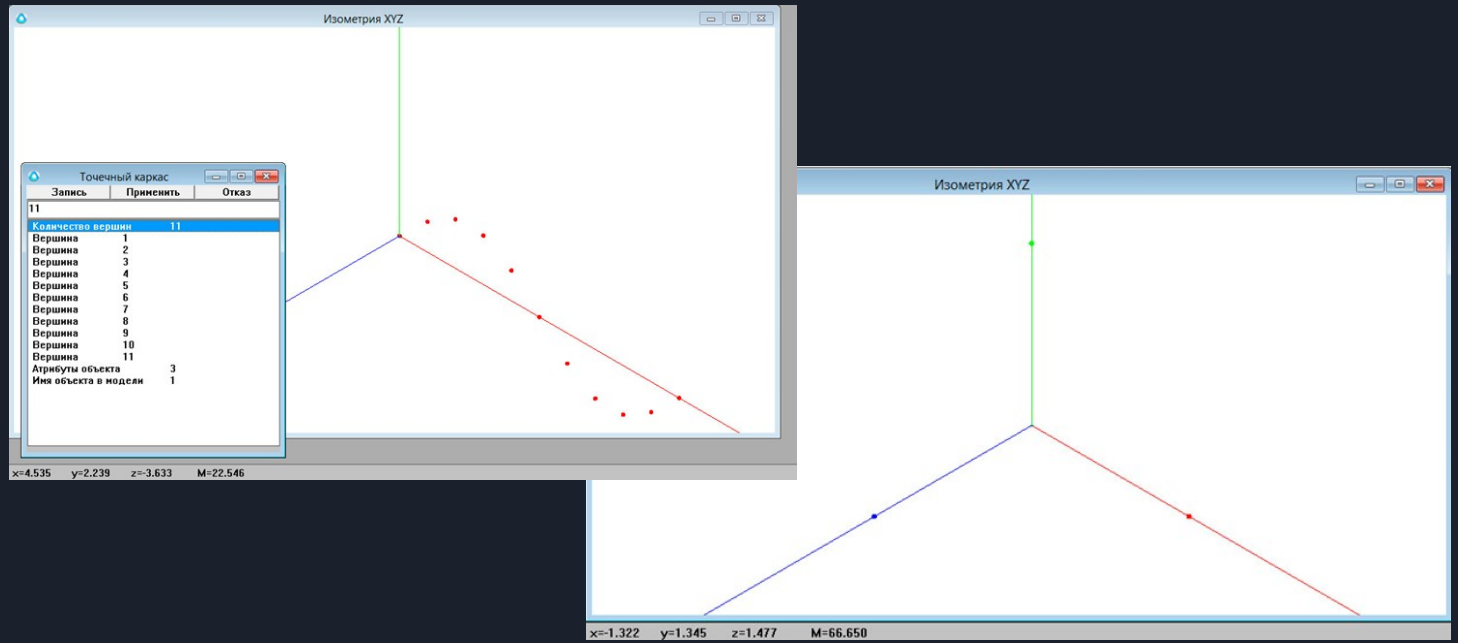
создание примеров использования функционала ядра C3D

Распределение ролей в проекте:

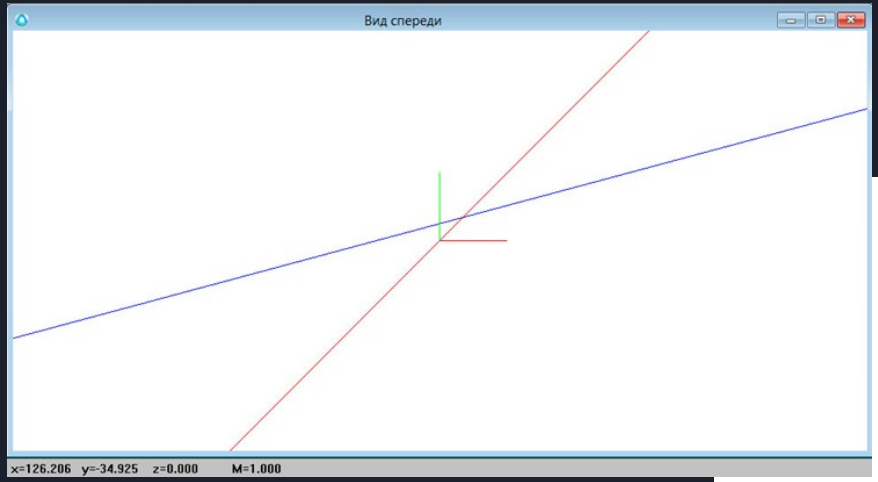
1)Серяков А.В. - разработка примеров, создание аннотаций;
2)Волобуев Р.А. - разработка примеров, создание аннотаций;
4)Фролов А.М. - создание сайта, создание презентации.

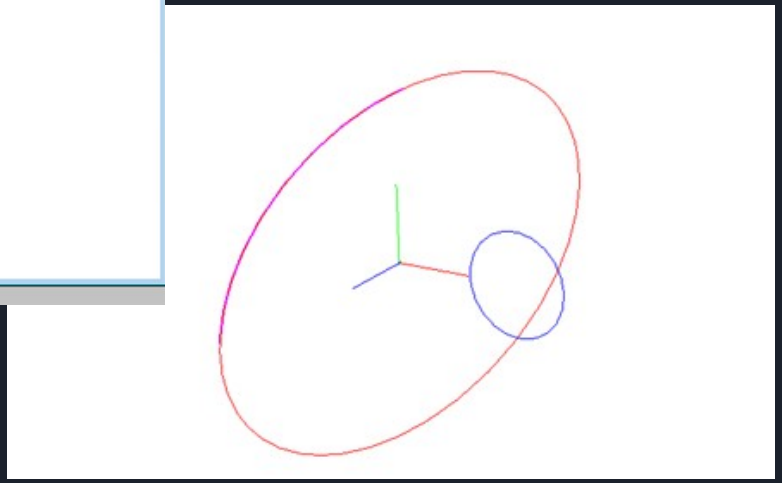
Этапы работы
1.
Определение базовых функций ядра

1.Построение точек в различных системах координат

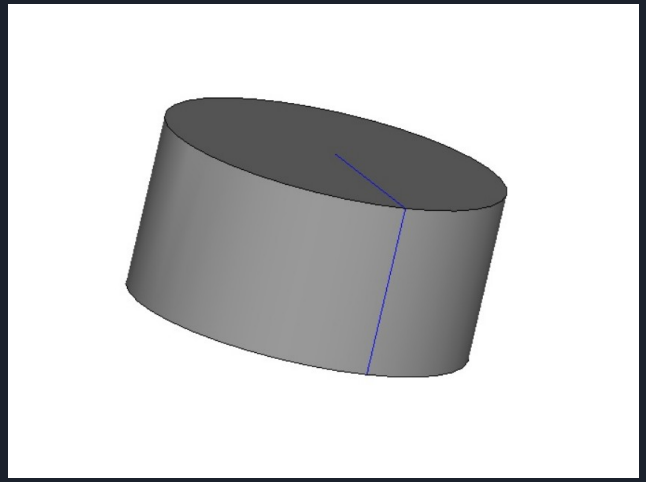


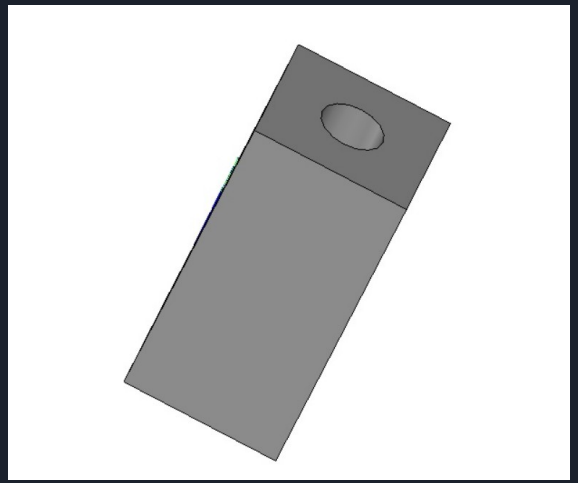
2.Построение прямых линий и кривых





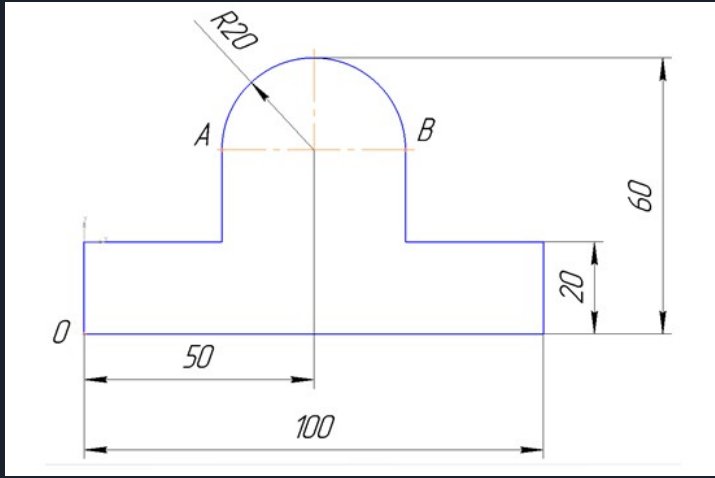
3.Построение поверхностей и тел вращения

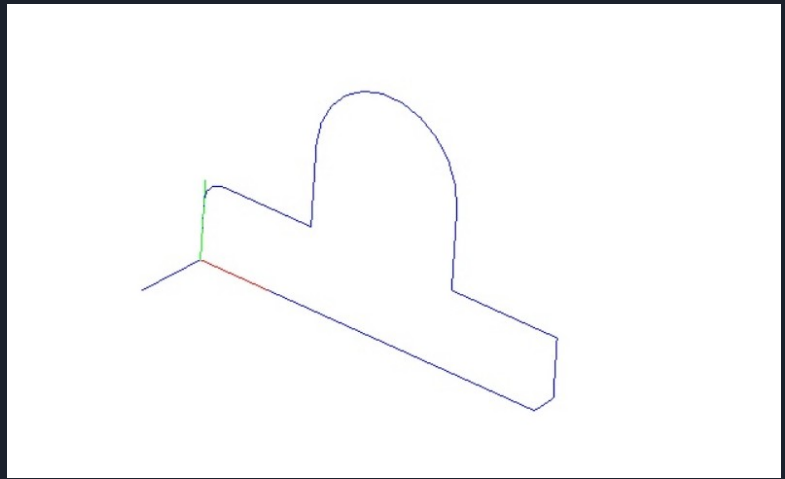




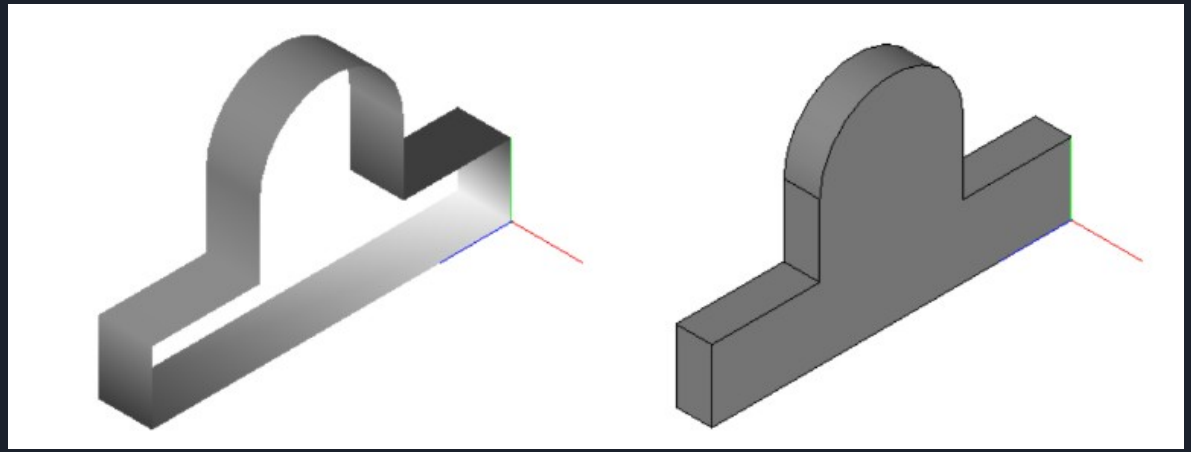
2.
Разработка программ для демонстрации функций ядра

1.Пример алгоритма по созданию эскиза

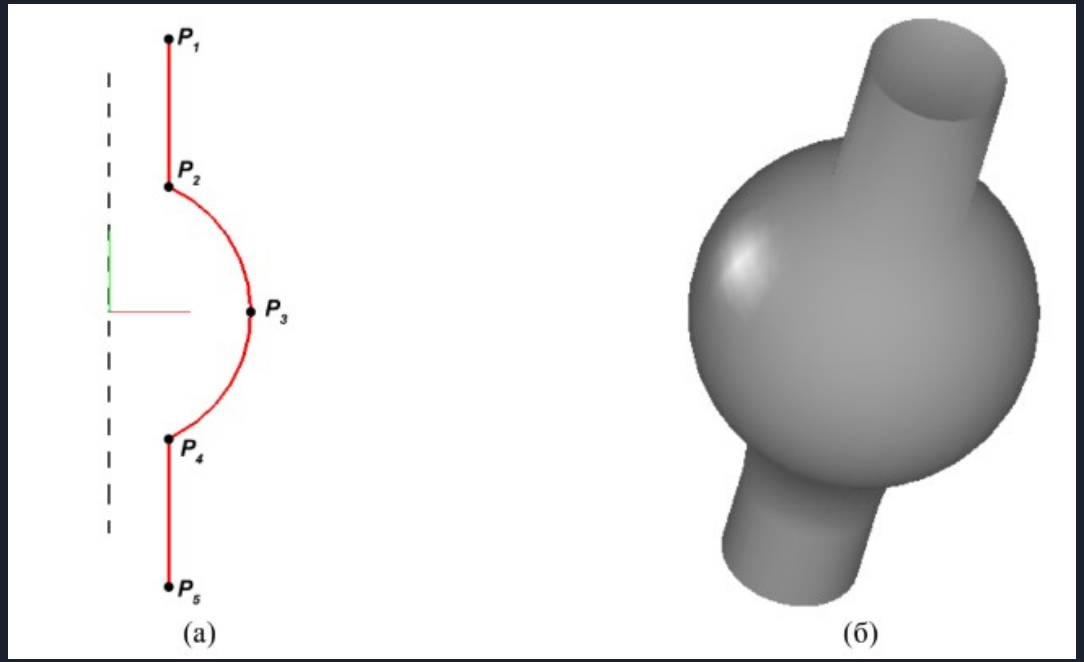




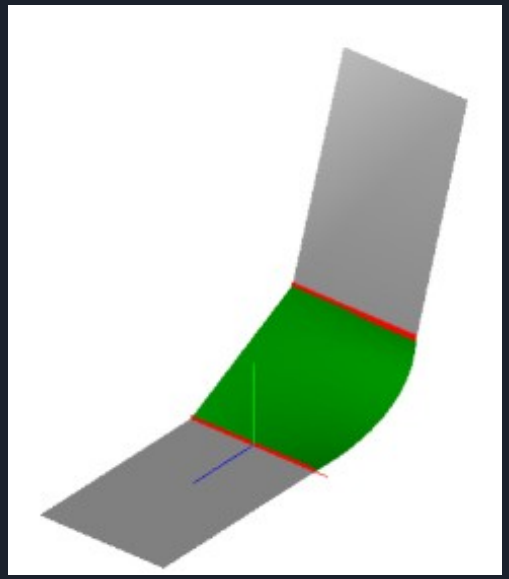
2.Пример алгоритма по созданию объемного тела путем выдавливания эскиза

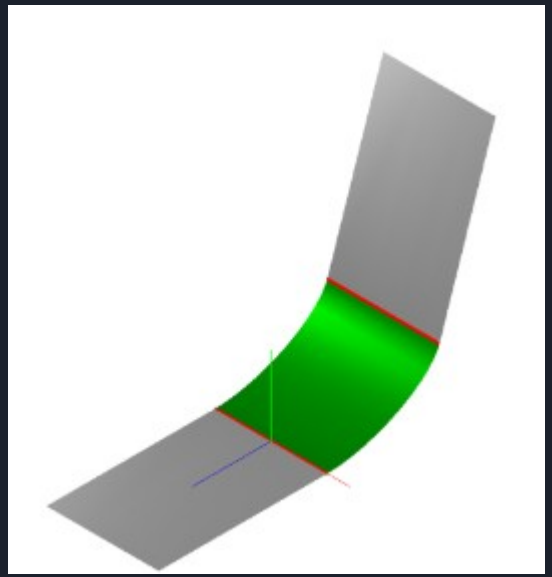


3.Пример алгоритма по созданию поверхностей путем вращения эскиза вокруг оси



4.Алгоритм создания кривых, фасок, сопряжений



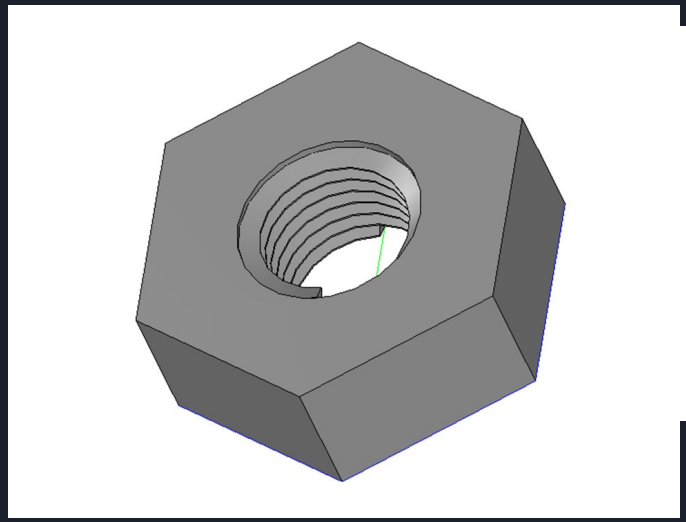


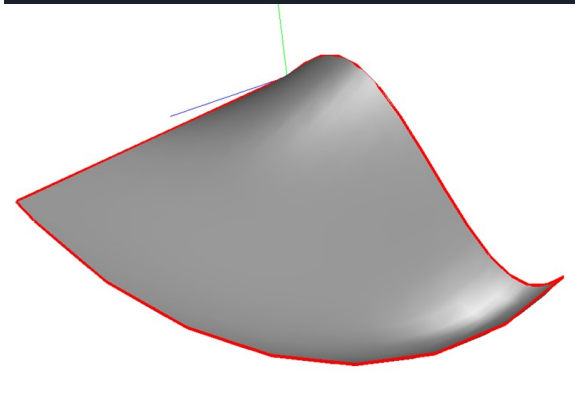
3.
Разработка структуры аннотации для программы

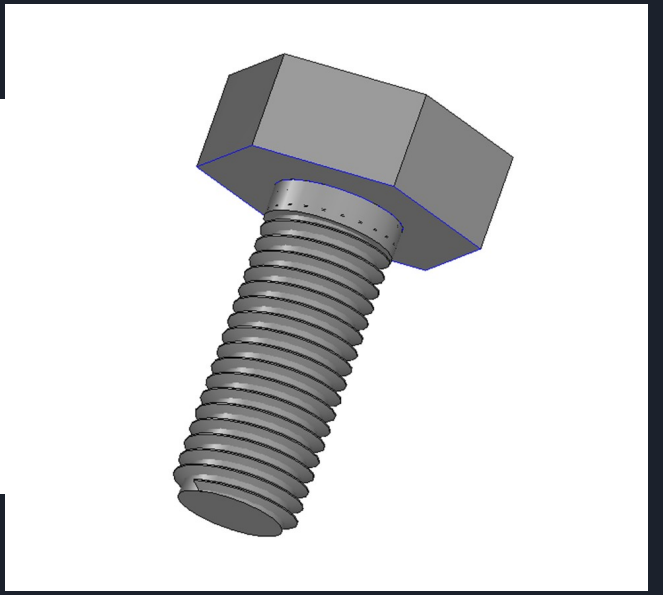
Каждая аннотация имеет следующее содержание:

1) Задача (постановка задачи, которую решает программа);
2) Поэтапный разбор кода программы с пояснениями;
3) Код программы, собранный воедино;
4) Список используемых в программе функций с пояснениями;
5) Список используемых в программе классов с пояснениями.

Примеры результатов работы







Примеры результатов работы

ПРИМЕР №4

Задача: реализация 3D модели гайки.

Поэтапное построение эскиза.

1. Создаем эскиз шестиугольника, описанного около окружно

```
void CreateSketch2(RPArray<MbContour>& _arrContours)
{
    // Центры и радиусы окружностей, дуги которых входят в контур
    const MbCartPoint centerCircle(0, 0);
    const double RAD = 20;
    // Количество сторон многоугольника
    const int SIDE_CNT = 6;
    // Радиус описывающей окружности многоугольника
    const double RAD1 = 45.0;

    // Массив для хранения вершин ломаной
    SArray<MbCartPoint> arrPnts(SIDE_CNT);
    // Вычисление вершин ломаной равномерным делением окружности
    for (int i = 0; i < SIDE_CNT; i++)
    {
        // Угловое положение i-й вершины на описывающей окружности.
        // Угловое положение начальной вершины - M_PI/2 (эта вершина
        // расположена на вертикальной оси).
        double angle = M_PI / 2 + 2 * M_PI / SIDE_CNT * i;
        MbCartPoint pnt(RAD1 * cos(angle), RAD1 * sin(angle));
        arrPnts.Add(pnt);
    }

    // Построение единой ломаной внешнего контура по точкам
    MbPolyline* pPolyline = new MbPolyline(arrPnts, true);
    MbContour* pContour = new MbContour(*pPolyline, true);

    // Построение окружности
    MbArc* pCircle = new MbArc(centerCircle, RAD);
    MbContour* pContourCircle = new MbContour(*pCircle, true);
    _arrContours.push_back(pContour);
    _arrContours.push_back(pContourCircle);
}
```

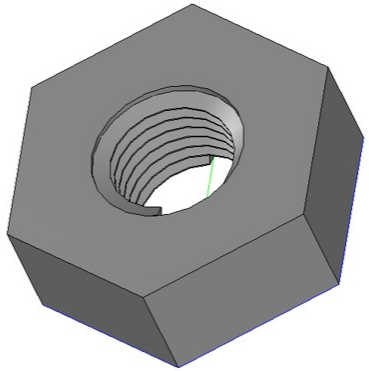
2. Построение эскиза резьбы:

```
void CreateSketch01(RPArray<MbContour>& _arrContours)
{
    // Параметры для эскиза
    double R = 18;
    double l = 2;

    // Массив точек для массив
    SArray<MbCartPoint>arrPnts(4);
    arrPnts.Add(MbCartPoint(-1 * 0.1 + l, -1 + R));
    arrPnts.Add(MbCartPoint(l * 0.1 + l, -1 + R));
    arrPnts.Add(MbCartPoint(2*l, l + R));
    arrPnts.Add(MbCartPoint(0, l + R));

    // Создание единого контура
    MbLineSegment* pS1 = new MbLineSegment(arrPnts[0], arrPnts[1]);
    MbContour* pContour = new MbContour(*pS1, true);

    MbLineSegment* pS2 = new MbLineSegment(arrPnts[1], arrPnts[2]);
    pContour->AddSegment(pS2);

    MbLineSegment* pS3 = new MbLineSegment(arrPnts[2], arrPnts[3]);
    pContour->AddSegment(pS3);

    MbLineSegment* pS4 = new MbLineSegment(arrPnts[3], arrPnts[0]);
    pContour->AddSegment(pS4);

    _arrContours.push_back(pContour);
}
```

3. Переходим к основному коду:

```
void MakeUserCommand0()
{
    // Локальная СК (по умолчанию совпадает с мировой СК)
    MbPlacement3D pl;

    // СОЗДАНИЕ КОНТУРОВ ДЛЯ ОБРАЗУЮЩЕЙ
    RPArray<MbContour> arrContours;
    CreateSketch2(arrContours);

    // Отображение образующей (в плоскости XY глобальной СК)
    for (int i = 0; i < arrContours.size(); i++)
        viewManager->AddObject(Style(1, RGB(0, 0, 255)), arrContours[i], &pl);

    // ПОСТРОЕНИЕ ТЕЛА ВЫДАВЛИВАНИЯ
    // Образующая размещается на плоскости XY глобальной СК.
    // Важное замечание: объект-плоскость должен создаваться динамически,
    // поскольку он продолжает использоваться в объекте-твердом теле после
    // выхода из данной функции.
    MbPlane* pPlaneXY = new MbPlane(MbCartPoint3D(0, 0, 0), MbCartPoint3D(1, 0, 0),
        MbCartPoint3D(0, 1, 0));

    // Объект, хранящий параметры образующей
    MbSweptData sweptData(*pPlaneXY, arrContours);

    // Направляющий вектор для операции выдавливания
    MbVector3D dir(0, 0, -1);

    // Параметры операции выдавливания, задающие свойства тела для построения:
    // расстояние выдавливания в прямом и в обратном направлении вдоль
    // направляющего вектора
    const double HEIGHT_FORWARD = 30.0, HEIGHT_BACKWARD = 0.0;
    ExtrusionValues extrusionParam(HEIGHT_FORWARD, HEIGHT_BACKWARD);

    // Служебный объект для именования элементов модели твердого тела
    MbSNameMaker operNames(ct_CurveExtrusionSolid, MbSNameMaker::i_SideNone, 0);
    RPArray<MbSNameMaker> cNames(0, 1, false);

    // Построение твердого тела выдавливания
    MbSolid* pSolid = NULL;
    MbResultType res = ::ExtrusionSolid(sweptData, dir, NULL, NULL, false,
        extrusionParam, operNames, cNames, pSolid);

    // Локальная СК (по умолчанию совпадает с мировой СК)
    MbPlacement3D pl1;

    // Построение направляющей кривой в виде незамкнутого NURBS-сплайна
    // 4-го порядка по контрольным точкам
    MbCurve3D* pSpiral = new MbCurveSpiral(MbCartPoint3D(0, 0, 0), MbCartPoint3D(0, 0,
-26), MbCartPoint3D(0, 26, 0), 4, false);

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

Результат построения.



Вывод

Данный проект нацелен на обеспечения удобства программистам при освоении ядра C3D путем создания примеров использования функций ядра.

СПАСИБО ЗА ВНИМАНИЕ