Мои представления о формате .nres

Что интересно, здесь все контуры дублируются в двух форматах точек, вот например начало файла, задание домена:

<количество контуров в принципе>

domain

<количество контуров в домене, допустим 2>

<количество точек в контуре №0, допустим 3>

<x,y - смещение, видимо для нестандартных контуров домена, у прямоугольника всегда по нулям>

<x,y,b (b всегда ноль)>

<x,y,b (b всегда ноль)>

<x,y,b (b всегда ноль)>

<количество точек в контуре №1, допустим 4>

<x,y>

<x,y,b (b всегда ноль)>

<x,y,b (b всегда ноль)>

<x,y,b (b всегда ноль)>

<x,y,b (b всегда ноль)>

<количество контуров в домене, допустим 2>

<количество точек в контуре №0, допустим 3>

<x,y,b>

<x,y,b>

<x,y,b>

<количество точек в контуре №1, допустим 4>

<x,y,b>

<x,y,b>

<x,y,b>

Подобным образом сохраняются далее и детали. Второй формат это тот самый формат точек который мы выводим из TFlex'а в таск для NF, стандартные XYB.

А вот первый более интересен, им я в итоге и воспользовался - во-первых всегда сначала идёт смещение XY, прямые линии (точки с нулевыми b) обозначаются тут так же, а вот дуги здесь обозначаются пятью аргументами:

X, Y, R, AngStart, AngSweep

, где X,Y - позиция первой точки дуги

R - радиус, описываемый дугой

AngStart - начальный угол дуги

AngSweep - угол, описываемый от начала дуги

Собственно про алгоритм - я сначала парился пытался сделать циклы последовательные, то есть читаю строку, цепляю из неё циферку 4, распознаю что это количество контуров, в ней для каждого контура так же цепляю циферку количества точек итд, но получалась фигня полная.

Сделал иначе - отталкиваясь от того что нам по сути по барабану какой контур откуда взялся, домен это или деталь, я просто начинаю построчно читать файл, разбивая строку функцией split по символу пробела. В случае если я натыкаюсь на строку с двумя аргументами – я понимаю, что попал на строку смещения, что означает начало нового контура. После этого я начинаю читать следующие строки, если у них 3 аргумента – я их добавляю в текущую полилинию, если 5 аргументов – я попал на дугу окружности, добавляю в полилинию точку начала дуги и закрываю полилинию, отрисовываю дугу и открываю полилинию заново, с точкой конца дуги. Если я при дальнейшем последовательном чтении натыкаюсь на строку с отличным от 3 или 5 количеством аргументов то я контур закрываю, замыкаю контур добавлением первой точки в последнюю полилинию и рисую его в TFlex-e, иду дальше по файлу, ищу опять строку с двумя аргументами, повторяю алгоритм и так до EOF.

Алгоритм отрисовки полилинии в принципе не сложен, кроме разве что того, что приходится пользоваться небезопасным кодом – почему-то полилинию я нашёл как задавать только с помощью указателей; в файле NFResult я написал класс NFPolyline для работы с полилинией, там есть методы добавления точки и метод отрисовки.

Алгоритм отрисовки дуг я сделал по трём точкам – точка начального угла, точка конечного угла и точка угла между ними:

double x1 **=** Math**.**Cos**(-**ang1 **/** 180 **\*** Math**.**PI**)** **\*** radius **+** x **+** radius**;**

double y1 **=** Math**.**Sin**(-**ang1 **/** 180 **\*** Math**.**PI**)** **\*** radius **+** y **+** radius**;**

double x2 **=** Math**.**Cos**((-**ang1 **-** ang2**/**2**)** **/** 180 **\*** Math**.**PI**)** **\*** radius **+** x **+** radius**;**

double y2 **=** Math**.**Sin**((-**ang1 **-** ang2**/**2**)** **/** 180 **\*** Math**.**PI**)** **\*** radius **+** y **+** radius**;**

double x3 **=** Math**.**Cos**((-**ang1 **-** ang2**)** **/** 180 **\*** Math**.**PI**)** **\*** radius **+** x **+** radius**;**

double y3 **=** Math**.**Sin**((-**ang1 **-** ang2**)** **/** 180 **\*** Math**.**PI**)** **\*** radius **+** y **+** radius**;**

где x и y – первые 2 аргумента, radius – аргумент №3, ang1 – начальный угол (аргумент №4), ang2 – угол, описываемый от начала дуги (аргумент №5)

Минусы выведены опять же эмпирически – методом научного тыка.

Из интересного ещё то, что ось Y у CatUI направлена вниз, а у TFlex’а вверх, таким образом я когда читаю из файла первый контур (а он всегда домен) подмечаю максимальный Y в переменную y\_offset, и в дальнейшем произвожу для всех координат Y перед непосредственной отрисовкой преобразование типа Yi = y\_offset – Yi;

И да, в конце файла есть информация по поводу занимаемой площади домена, раскроя и их соотношение, но ею я не пользовался, подумал что в рамках плагина это не нужно, пользователь и в CatUI может посмотреть результат.