Разработка программного приложения с графическим вводом данных. Язык C++ (разработка на QT приветствуется).

Доброго всем дня! Перейду сразу к заданию.

Имеется Алгоритм вычислительной части для программы и предполагаемый концепт программного приложения:

Входные и выходные данные представлены в описании.

Алгоритм программы изложен в виде очень понятного псевдокода.

Вся работа заключается в создании панели для рисования простейших схем (линии и точки) и визуализации работы предоставленного алгоритма.

Для знающего программиста выполнение работы не должно занять много времени.

В работе мне очень важен качественный и понятный код, желательно ООП т.к. планируется дальнейшая доработка и развитие темы.

Цена за работу очень зависит от качества и детальности выполненной работы, т.е. чем больше усилий будет приложено, тем больше я готов заплатить за работу. Готов ознакомится с любым видением выполненной работы и обсудить оплату. Также готов ответить на любые вопросы в части работы, если есть непонимание.

Спасибо!

Описание алгоритма и интерфейса.

Суть работы - программная реализация метода вычисления позиции точки (решение обратной задачи кинематики). Есть произвольная система вершин и ребер (Кинематический граф), которая является представлением робота. Пример:

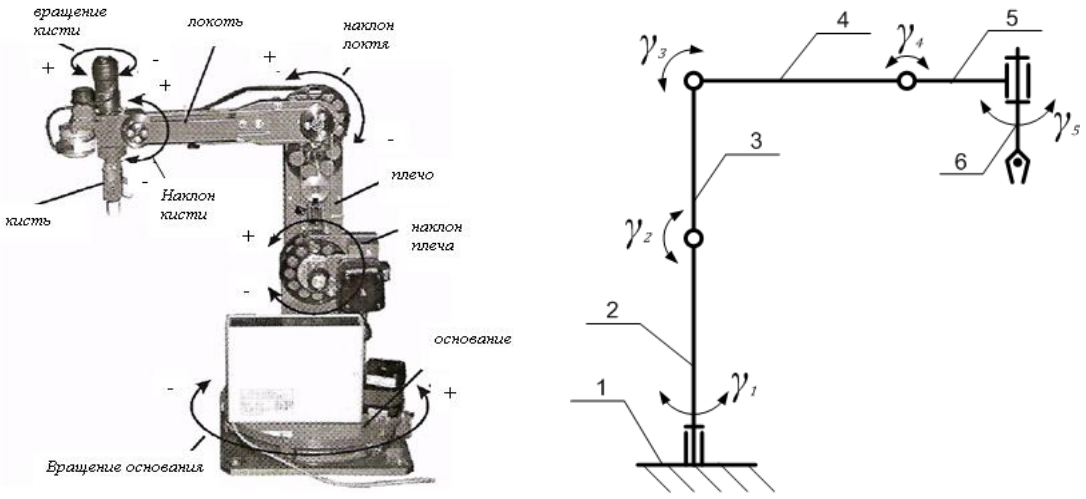

Рис.1

Как видно, система состоит из сочленений и звеньев. Метод, который нужно запрограммировать, позволяет определить положение системы при заданных координатах точки (скажем схвата робота). Т.е. исходными данными являются декартовы координаты точки, а на выходе нужно получить углы поворота в сочленениях, чтобы схват робота оказался в заданной точке.

Метод довольно простой, алгоритм его следующий (поэтапно):

Данные ввода: Задаем позицию (координаты) всех точек (вершин) $pi,\ i = 1, \ldots, n$, координаты целевого конечного положения t (на рисунке 2 - p'_4), а также длины всех ребер, которые можно вычислить по формуле:

$$d_i = |p_{i+1} - p_i|, \text{где } i = 1, \ldots, n-1.$$

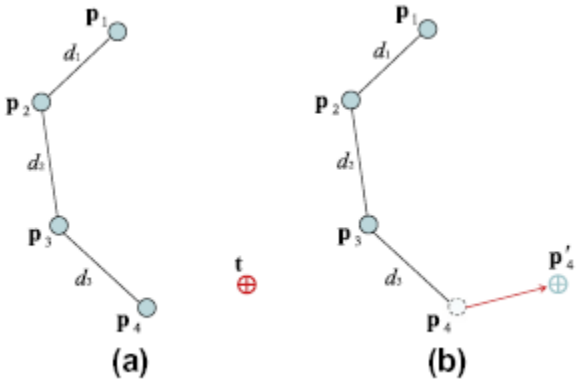

Рис.2

Данные вывода: Новые позиции вершин $p_i,\ i = 1,\ldots,n$ (точнее их координаты).

1. //Вычисляем расстояние между первой вершиной (корневой точкой) и целевой точкой:
2. $dist\ =\ |p_1\ -\ t|$
3. // проверяем возможность достижения цели
4. $if\ dist\ >\ d_1\ +\ d_2 + \ldots + d_{n-1}\ then$
5. // Цель не достижима
6. $for\ i = 1,\ldots,n-1\ do$
7. // Определяем расстояние ri между целью t и вершиной графа p_i
8. $r_i = |t - p_i|$
9. $\beta_i = d_i/r_i$
10. // вычисляем новую позицию для каждого сочленения p_i
11. $p_i + 1 = (1 - \beta_i)\, p_i\ +\ \beta_i t$
12. end;
13. else
14. // цель достижима. Запоминаем позицию первого сочленения p_1
15. $base\ =\ p_1$
16. // Проверяем, приблизилась ли конечная вершина графа pn к целевой точке t на допустимое расстояние
17. $diff\ =\ |p_n - t|$
18. $while\ diff\ >\ aim$ // aim - допустимое расстояние вокруг цели, является исходными данными
19. // Этап 1 - прямое движение (рис.3)
20. // Устанавливаем конечную позицию pn в цель t
21. $p_n = t$
22. $for\ i = n-1,\ldots,1\ \ do$
23. // Определяем расстояние r_i между новой позицией сочленения $p_i + 1$ и сочленения pi
24. $r_i = |p_{i+1}\ -\ p_i|$
25. $\beta_i\ =\ d_i/r_i$
26. // Определяем новую позицию p_i
27. $p_i = (1 - \beta_i)\, p_{i+1}\ +\ \beta_i p_i$

28. end

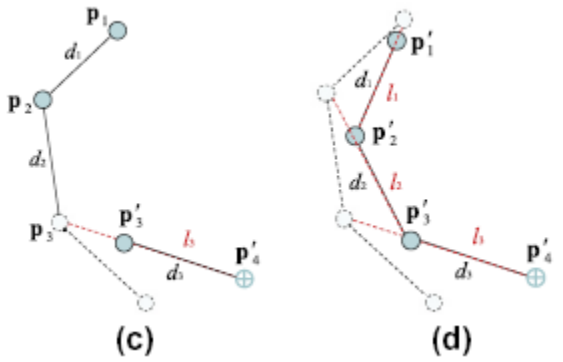

Рис.3

29. // Этап 2 - обратное движение (рис.4)
30. // Устанавливаем корневую позицию p_1 в цель ранее заданную базовую точку base
31. $p_1 = base$
32. $for\ i = 1, \ldots, n-1\ do$
33. // Определяем расстояние ri между новой позицией сочленения p_{i+1} и сочленения p_i
34. $r_i = |p_{i+1} - p_i|$
35. $\beta_i = d_i / r_i$
36. // Определяем новую позицию p_{i+1}
37. $p_{i+1} = (1 - \beta_i)\, p_i + \beta_i p_{i+1}$
38. end
39. $diff = |p_n - t|$
40. end
41. end

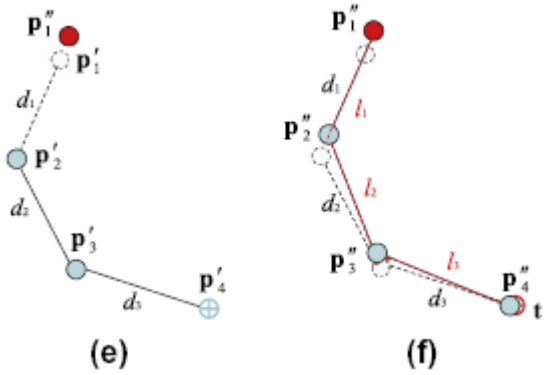

Рис.4

Таким образом, входные данные - это координаты всех точек механической системы $pi,\ i = 1, \ldots, n.$

А на выходе программа должна выдавать новые координаты всех точек при условии, что система перешла в состояние $p_n = t + aim$ (aim - допустимая погрешность) а расстояния между точкам ($d_i = |p_{i+1} - p_i|$) и остались прежними.

Интерфейс у программы должен быть следующий:

1. На главной форме размещается палитра (подобно программе paint или ms visio) с сеткой.

2. С помощью набора инструментов на палитре размещаются точки (сочленения), их координаты необходимо вводить в ручную. Далее устанавливаются связи между точками (звенья). Затем определяем целевую позицию.

3. В ходе расчета по указанному выше алгоритму система должна переходить в целевую точку.

4. Процесс перехода к цели должен быть выполнен в виде анимации с возможностью замедления (чтобы наглядно видеть работу алгоритма).

Большая просьба писать комментарии, т.к. в дальнейшем планируется доработать программу для выполнения указанных действий в 3-х мерном пространстве.