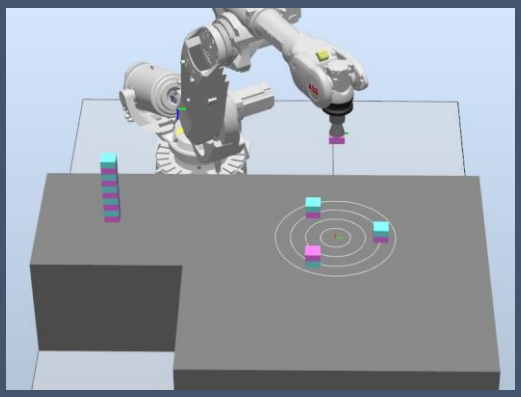


Memoria 2

Traslado de piezas en bloque

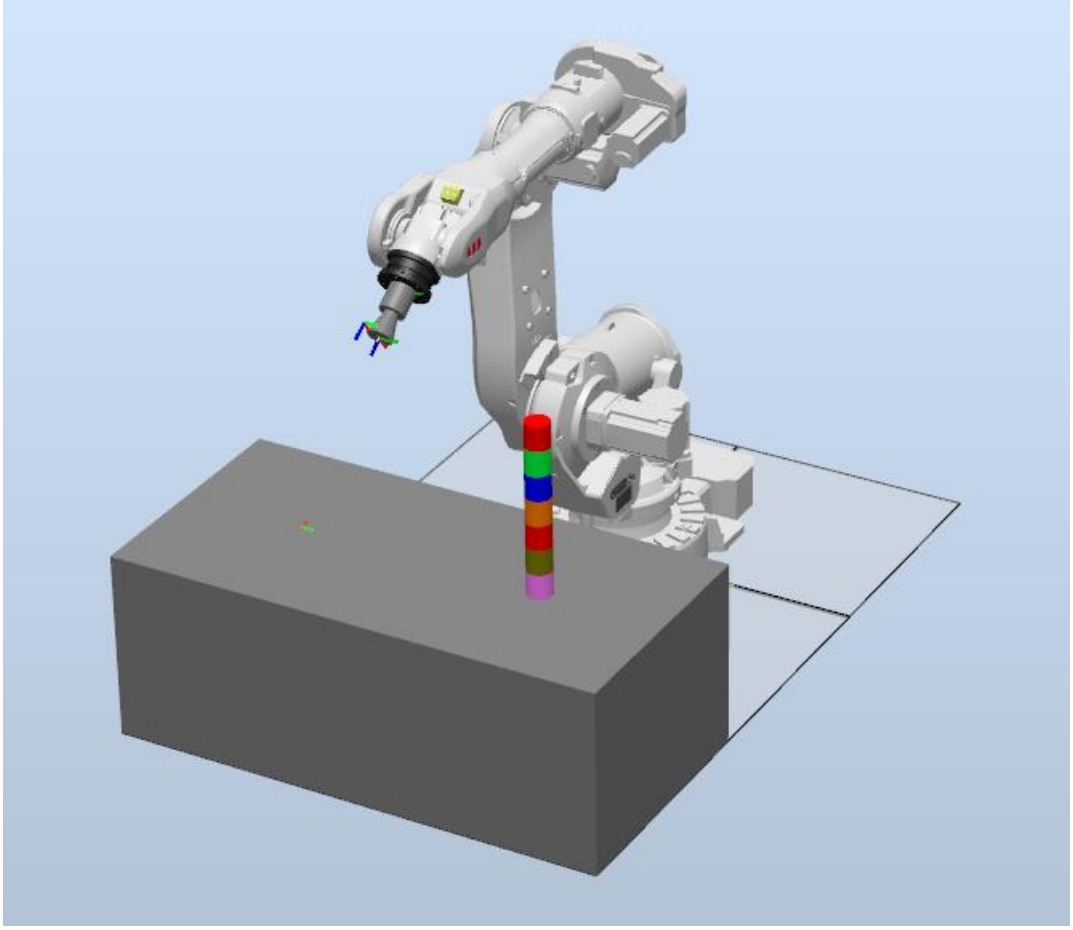
Francisco José Caballero del Campo

Tabla de contenido

Versión básica 2
Estación de partida 2
Ejercicio básico 2
Simulaciones 4
Simulación con 7 piezas 4
Simulación con 4 piezas 4
Versión avanzada (I) 6
Estación de partida 6
Ejercicio Avanzado 1 6
Simulaciones 9
Simulación con 10 piezas y 4 torres con radio 300 9
Simulación con 17 piezas y 3 torres con radio 200 9
Simulación 10 piezas y 4 torres con radio 300 10
Versión Avanzada (II) 11
Estación de partida 11
Ejercicio avanzado 2 11
Simulaciones 13
Simulación 3 piezas 13

Versión básica

Estación de partida



Montaje según lo hecho en clase.

Ejercicio básico

Para la realización de este ejercicio, debemos llevar la torre del punto origen al punto destino moviendo las piezas de una en una y respetando el orden inicial. Para ellos, necesitamos montar otra torre en un punto intermedio donde las piezas estarán en orden inverso al deseado y desde ese punto intermedio, moverlas de una en una al punto destino, donde tendrá el orden deseado.

El código Rapid realizado para este ejercicio consta de una función llamada MueveTorre cuyos parámetros de entrada son: un punto de origen, un punto destino, el número de piezas de la torre origen y un número para el desplazamiento, este desplazamiento sirve para mover la posición en el eje Y el punto donde se encontrará la torre intermedia, su valor se contará según el número de diámetros que queramos desplazar las torres.

Dicha función esta formada principalmente por dos bucles FOR anidados, el primero de ellos sirve para recorrer cada uno de los puntos de la torre, es decir, recorre una variable llamada “tabPos{3}” con tres puntos dentro, siendo la posición uno el punto origen, la posición 2 el punto intermedio y la posición 3 el punto destino, de esta forma en la primera iteración solo accedemos a los puntos origen e intermedio y en la segunda a los puntos intermedio y destino; y el segundo bucle FOR sirve para recorrer todos los puntos de las piezas que conforman la torre “origen” para llevarlas al “destino” activando y desactivando la ventosa.

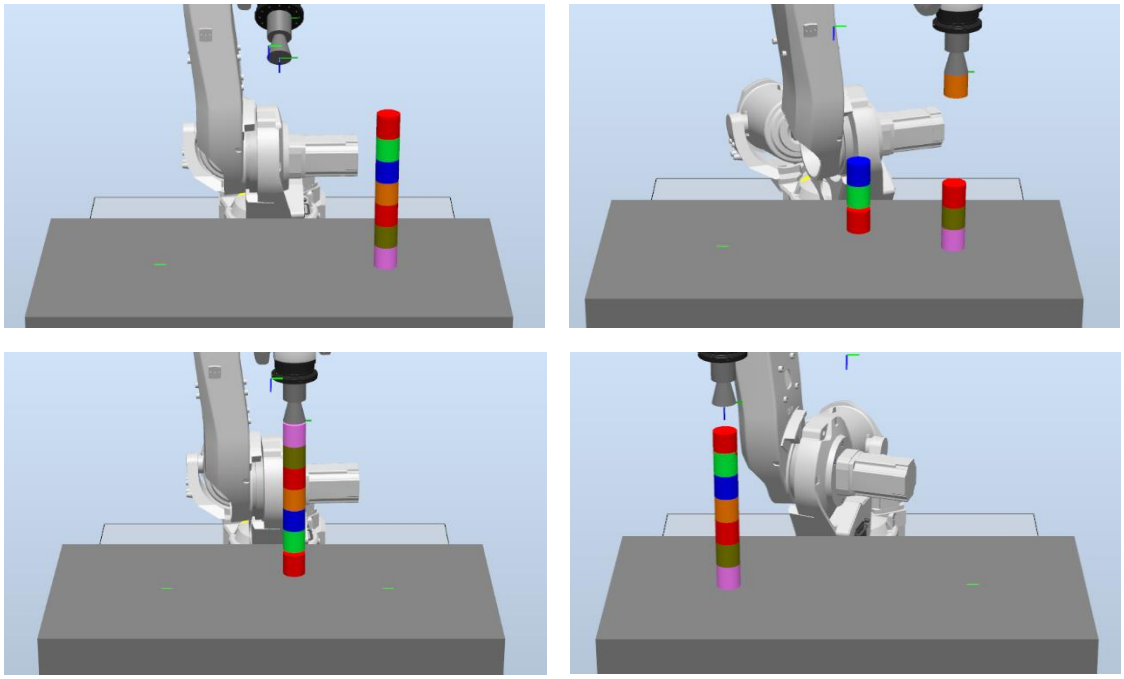
El código Rapid es el siguiente:

```
PROC main()
    ! Movemos la primera torre de 7 piezas 2 veces el diametro de la pieza
    MueveTorre pBaseTorre1, pBaseTorre2, 7, 4;

ENDPROC

PROC MueveTorre(robtarget pOrigen,robtarget pDestino, num nPiezas, num desplazamiento)
   !Array donde vamos a guardar todos los puntos para usarlos en un bucle FOR
   VAR robtarget tabPos{3}; !Punto inicial, intermedio, y final
   VAR robtarget pIntermedio;
   VAR robtarget pAproxMonton1;
   VAR robtarget pAproxMonton2;
   VAR robtarget pMonton1;
   VAR robtarget pMonton2;

   !Creamos el punto intermedio con la altura unicial
   pIntermedio := Offs (pOrigen, 0, diametro * desplazamiento * (-1), altura);

   !Añadimos los 3 puntos a la tabla
   tabPos{1} := pOrigen;
   tabPos{2} := pIntermedio;
   tabPos{3} := pDestino;

   !Creamos el bucle FOR
   FOR i FROM 1 TO 2 DO !Recorreremos el bucle primero con los puntos i e i+1 y despues con i+1 e i+2

   !Definimos los puntos iniciales de los dos montones
   pMonton1 := Offs (tabPos{i}, 0, 0, altura * nPiezas);
   pMonton2 := Offs (tabPos{i+1}, 0, 0, altura);

    !Creamos los 3 puntos de aproximación
   pAproxMonton1 := Offs (tabPos{i}, 0, 0, (altura * nPiezas) + altura);
   pAproxmonton2 := Offs (tabPos{i+1}, 0, 0, (altura * nPiezas) + altura);

    FOR nPiezas2 FROM 1 TO nPiezas DO !Bucle para el número de piezas

        !Movemos el brazo a la posición de aproximación del monton 1
        MoveL pAproxMonton1, v2000, fine, Ventosa\WObj:=wobj0;

        !Cogemos la pieza del monton 1
        MoveLDO pMonton1, v2000, fine, Ventosa\WObj:=wobj0, SD_ActivaVentosa, 1;
        WaitDI ED_PiezaCogida, 1; !Esperamos a que PiezaCogida esté a 1
        MoveL pAproxMonton1, v2000, fine, Ventosa\WObj:=wobj0;

        !Movemos el brazo a la posicion de aproximación del monton 2
        MoveL pAproxmonton2, v2000, fine, Ventosa\WObj:=wobj0;

        !Ponemos la pueza en el montón 2
        MoveLDO pMonton2, v2000, fine, Ventosa\WObj:=wobj0, SD_ActivaVentosa, 0;
        WaitDI ED_PiezaCogida, 0; !Esperamos a que PiezaCogida esté a 1
```

```
61          MoveL pAproxmonton2, v2000, fine, Ventosa\WObj:=wobj0;
62
63          !Calculamos los nuevos puntos de cada montón
64          pMonton1 := Offs (pMonton1, 0, 0, - altura);
65          pMonton2 := Offs (pMonton2, 0, 0, altura);
66
67      ENDFOR
68    ENDFOR
69
70
71
72  ENDPROC
```

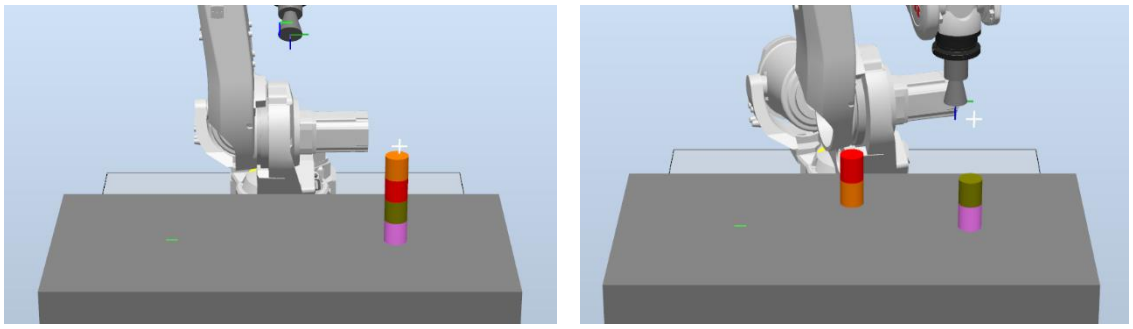
Simulaciones

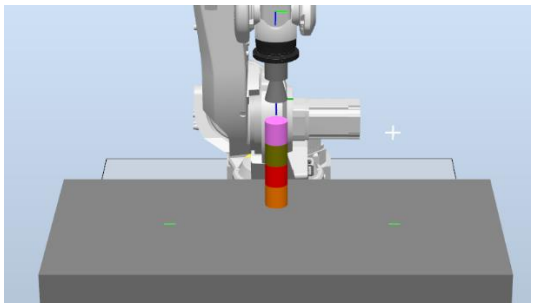
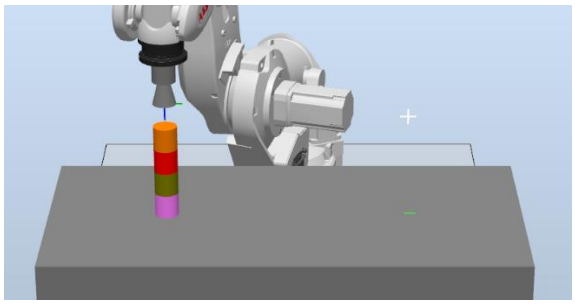
Simulación con 7 piezas



Simulación con 7 piezas en la posición origen y un desplazamiento de 4.

Simulación con 4 piezas

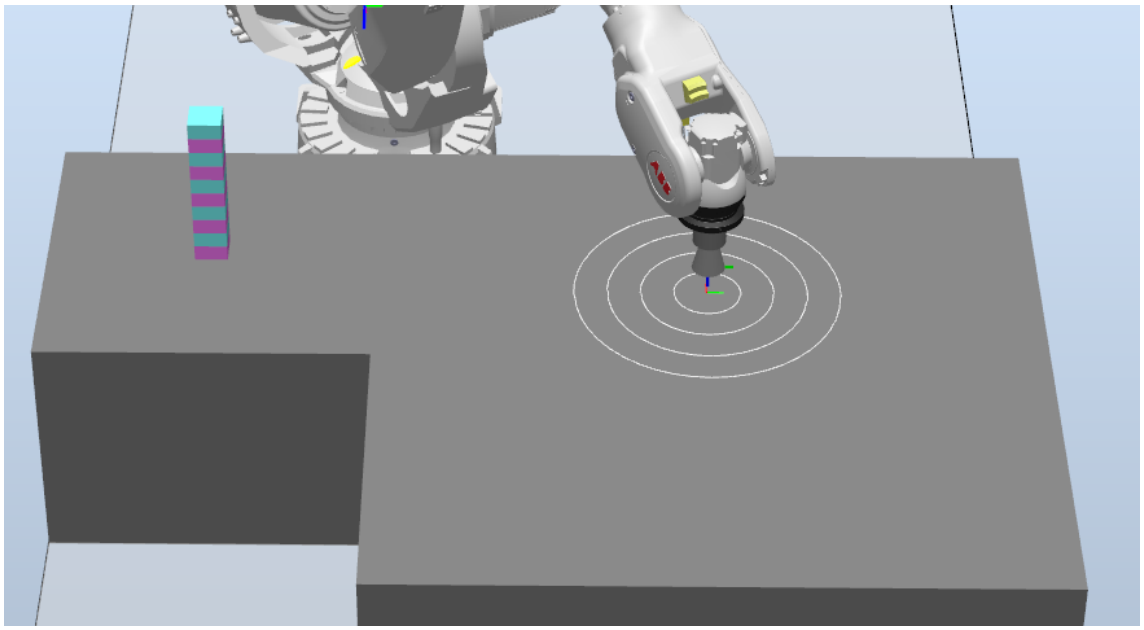


Simulación con 4 piezas en la posición origen y desplazamiento 5.

Versión avanzada (I)

Estación de partida



Estación de partida dada por el profesor.

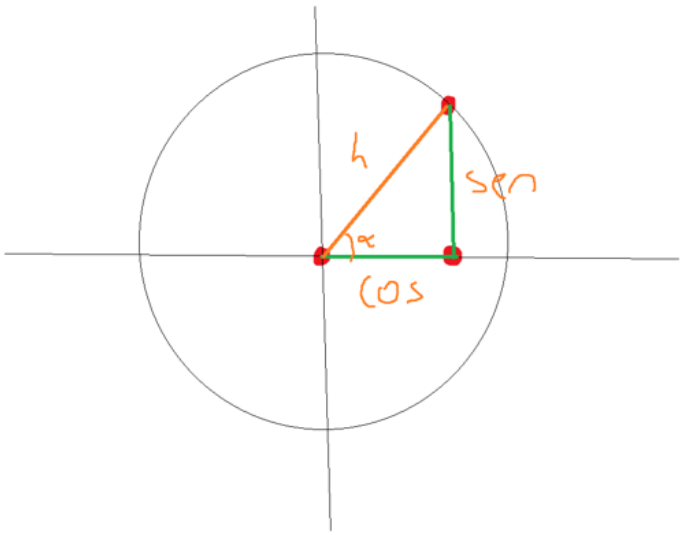
Consta de una torre de diez piezas apiladas en una posición origen y un punto destino con círculos concéntricos con radios de 100-200-300-400 milímetros.

Ejercicio Avanzado 1

Para la realización de este ejercicio tenemos que mover las piezas de la torre origen de tal manera que proporcionando un radio y un numero de torres en el destino, dichas piezas formen torres más pequeñas en el circulo con el radio proporcionado y separadas entre si la misma distancia con una altura equivalente, en el caso de que se pueda.

El código Rapid de este ejercicio consta de una función Ejercicio2 cuyos parámetros de entrada son: numero de piezas en el origen, numero de torres y radio de la circunferencia.

Esta función está formada por tres bucles FOR, en el primero calculamos y guardamos en un “array” los puntos donde se ubicarán las torres en el circulo seleccionado. Cogiendo como referencia el punto destino, calculamos la posición en el eje X haciendo uso del seno y la posición en el eje Y haciendo uso del coseno.



sen(a) = cat. opuesto / hipotenusa

cos(a) = cat. contiguo / hipotenusa

hipotenusa = radio

Los otros bucles FOR son dos bucles anidados de tal forma que en el primero vamos aumentando la altura de las torres destino cuando estás tengan la misma altura y en el segundo bucle, vamos recorriendo la torre origen y poniendo las piezas en las torres que se encuentran en el círculo, añadimos también una restricción para que en el caso de que se acaben las piezas en el origen y no tengan todas las torres destino la misma altura, el brazo se quede en la posición de reposo.

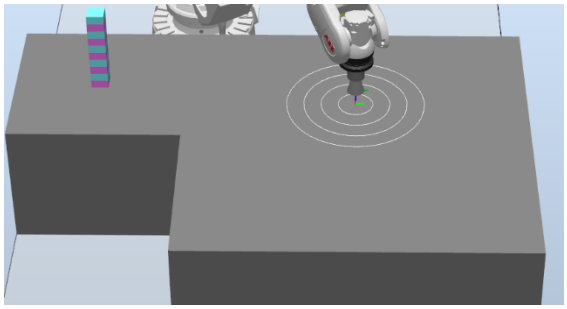
El código Rapid es el siguiente:

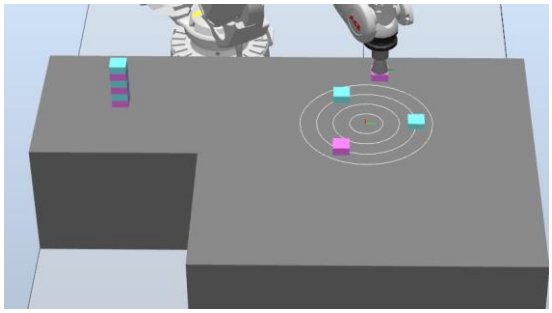
```
11  PROC main()
12      Ejercicio2 10, numeroDeTorresDestino, 300;
13  ENDPROC
14
15  PROC Ejercicio2 (num nPiezasTorreOrigen, num nTorresDestino, num radioCircunferencia)
16      VAR robtarget tabPos{numeroDeTorresDestino};
17      VAR robtarget pAproxCirculo;
18      VAR robtarget pAproxOrigen;
19      VAR robtarget pBase;
20      VAR robtarget pAux;
21      VAR robtarget pApoyo;
22      VAR num gradosIncremt := 0;
23      VAR num grados := 0;
24      VAR num nPiezasEnTorre := 0;
25      VAR num alturaTorreDestino := 0;
26      VAR num alturaTorreOrigen := 0;
27      VAR num X := 0;
28      VAR num Y := 0;
29      VAR num hipotenusa;
30
31      !Inicializamos
32      alturaTorreOrigen := altura * nPiezasTorreOrigen;
33
34      !Vamos a crear un punto de aproximación encima del centro del circulo y 5 piezas por encima de la torre origen
35      pAproxCirculo := Offs (pCentroDestino, 0, 0, altura * nPiezasTorreOrigen);
36      pAproxOrigen := Offs (pBaseOrigen, 0, 0, (nPiezasTorreOrigen * altura) + (altura * 5));
```

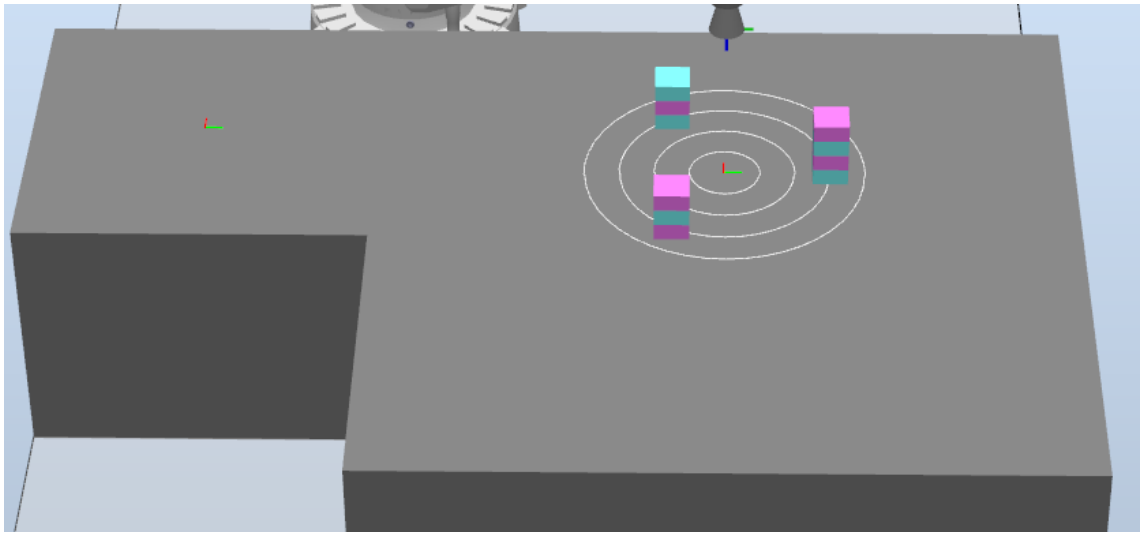
```
37
38          !Creamos el punto inicial de la torre origen
39          pBase := Offs (pBaseOrigen, 0, 0, altura * nPiezasTorreOrigen);
40
41          !Como el círculo tiene 360 grados, podemos descomponen los grados en tantas partes como puntos tenemos
42          gradosIncremt := 360 / nTorresDestino;
43          alturaTorreDestino := altura;
44          hipotenusa := radioCircunferencia;
45
46          !Vamos a buscar las posiciones de las torres destino
47          FOR i FROM 1 TO nTorresDestino DO
48
49              !Calculamos las posiciones haciendo uso de senos y cosenos
50              X := Sin(grados) * hipotenusa;
51              Y := Cos(grados) * hipotenusa;
52
53              !Creamos los puntos
54              pAux := Offs (pCentroDestino, X, Y, 0);
55
56              !Lo añadimos en la tabla
57              tabPos{i} := pAux;
58
59              !Actualizamos los grados
60              grados := grados + gradosIncremt;
61
62          ENDFOR
63
64          !En este segundo bucle for recorremos todos los puntos del array cogiendo y dejando las piezas
65          FOR A FROM 1 TO nPiezasTorreOrigen DO
66              FOR B FROM 1 TO nTorresDestino DO
67
68                  !Restriccion para que una vez se acaben las piezas no haga nada
69                  IF alturaTorreOrigen <= 0 THEN
70                      Break;
71                  ELSE
72
73                  !Movemos al punto de aproximación de la Torre Origen
74                  MoveL pAproxOrigen, v2000, fine, Ventosa\WObj:=wobj0;
75
76                  !Cogemos la pieza de la Torre Origen
77                  MoveLDO pBase, v2000, fine, Ventosa\WObj:=wobj0, SD_ActivaVentosa, 1;
78                  WaitDI ED_PiezaCogida, 1;
79                  MoveL pAproxOrigen, v2000, fine, Ventosa\WObj:=wobj0;
80
81                  !Movemos al punto de aproximacion del centro del círculo;
82                  MoveL pAproxCirculo, v2000, fine, Ventosa\WObj:=wobj0;
83
84                  !Depositamos la pieza en su sitio
85                  pApoyo := Offs (tabPos{B}, 0, 0, alturaTorreDestino);
86                  MoveLDO pApoyo, v2000, fine, Ventosa\WObj:=wobj0, SD_ActivaVentosa, 0;
87                  WaitDI ED_PiezaCogida, 0;
88                  MoveL pAproxCirculo, v2000, fine, Ventosa\WObj:=wobj0;
89
90                  !Bajamos la altura de la torre Base
91                  pBase := Offs (pBase, 0, 0, -altura);
92
93                  ENDIF
94
95                  !Decrementamos la altura en la torre origen para poder usar la restricción de altura
96                  alturaTorreOrigen := alturaTorreOrigen - altura;
97
98              ENDFOR
99
100             !Cuando colocamos todas las torres con la misma altura la actualizamos
101             alturaTorreDestino := alturaTorreDestino + altura;
102
103         ENDFOR
104
105     ENDPROC
```

Simulaciones

Simulación con 10 piezas y 4 torres con radio 300

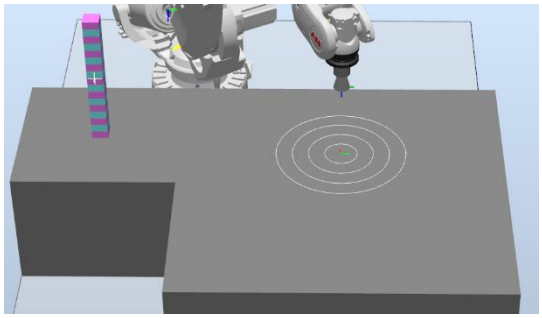


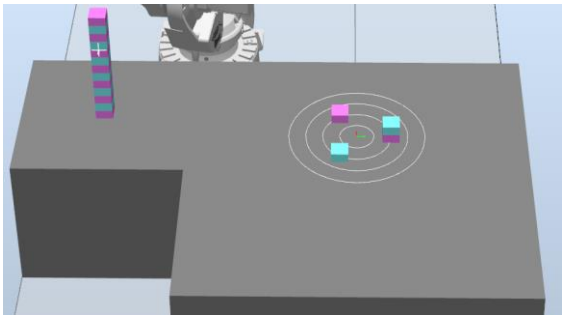


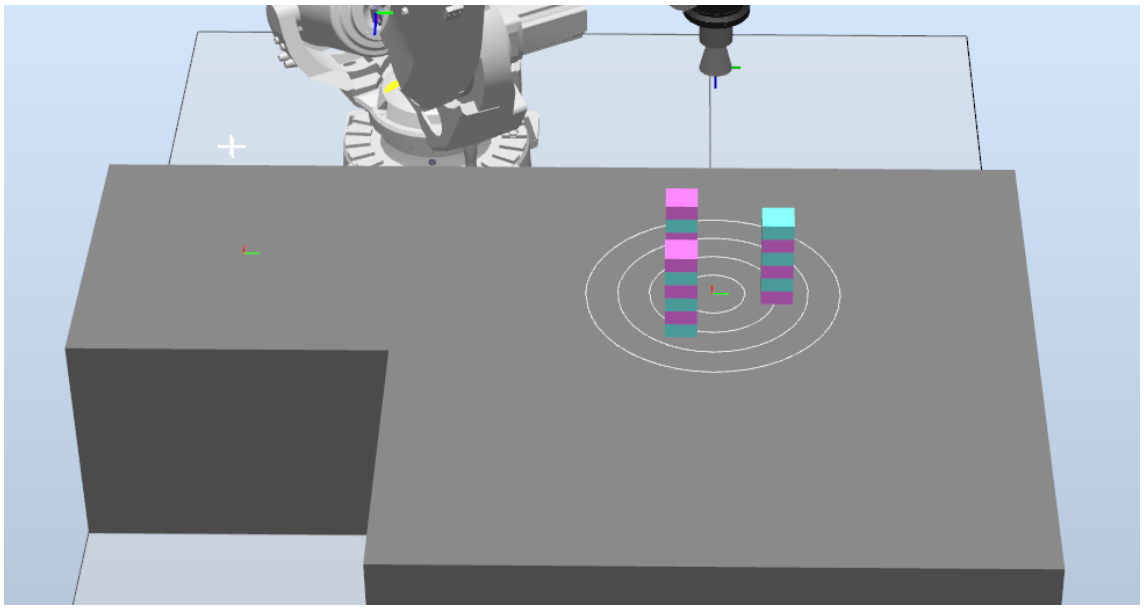


Podemos observar que la altura de las torres en el destino es diferente y el brazo acaba en la posición de reposo.

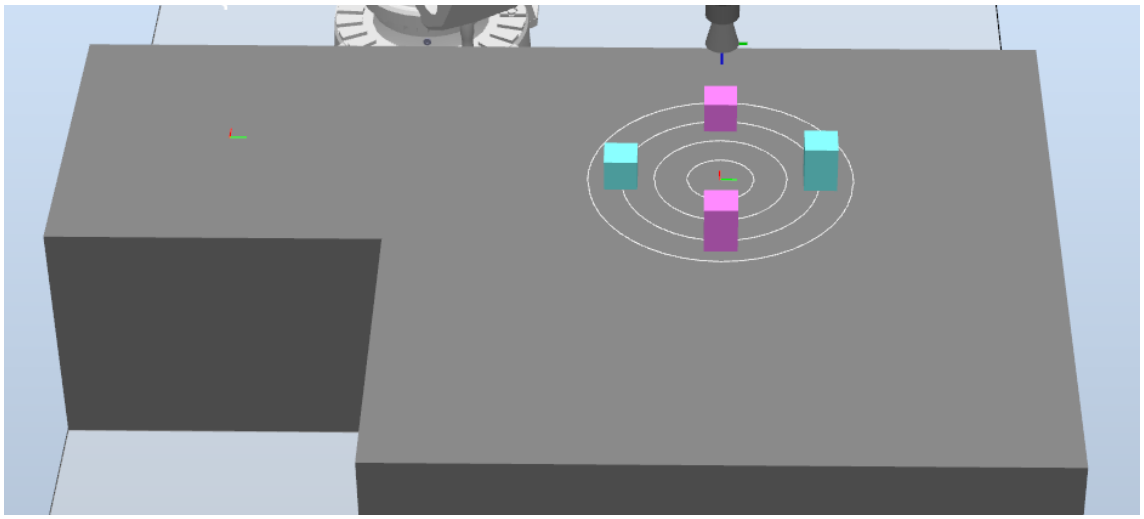
Simulación con 17 piezas y 3 torres con radio 200





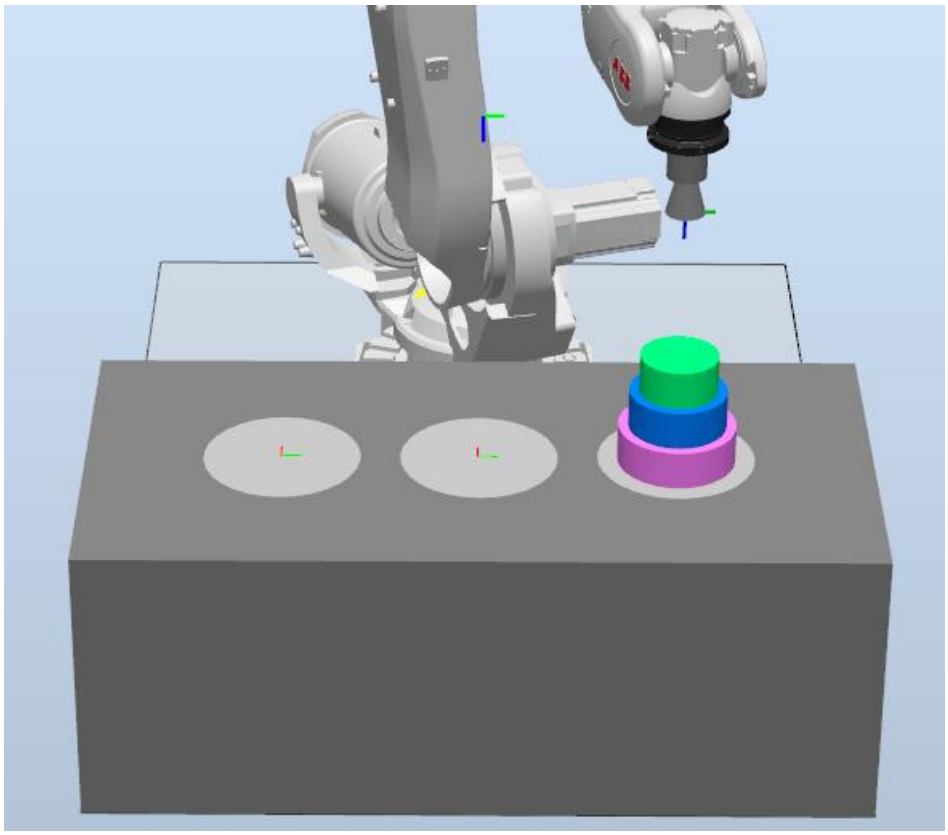


Simulación 10 piezas y 4 torres con radio 300



Versión Avanzada (II)

Estación de partida



Estación de partida proporcionada por el profesor.

Ejercicio avanzado 2

Este ejercicio trata de resolver el problema de las torres de Hanoi, es decir, mover la torre existente de una posición a otra, moviendo las piezas de una en una con la restricción de que no se puede poner una pieza encima de otra más pequeña.

Este ejercicio no he sido capaz de resolverlo ya que me ha sido imposible calculas a que altura se encuentra cada pieza en cada iteración.

En el código Rapid tenemos una llamada a la función TorresHanoi cuyas entradas son: numero de piezas, punto origen, punto destino, punto intermedio y altura de la pieza.

Dentro de la función tenemos una expresión IF-ELSE la cual va llamándose de manera recursiva.

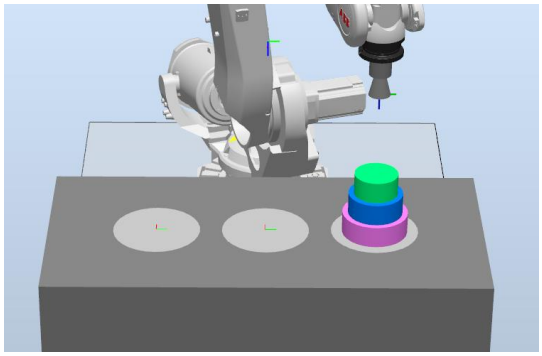
El código Rapid es el siguiente:

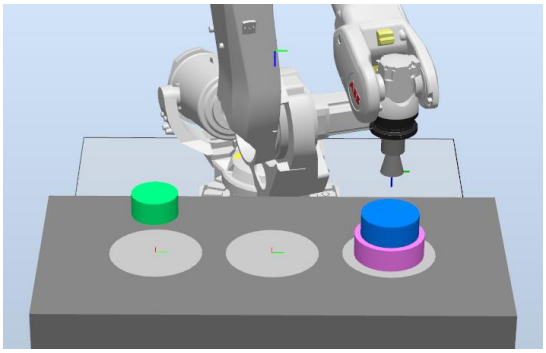
```
16  PROC main()
17      VAR robtarget pOrig;
18      pOrig := Offs (pBaseOrigen, 0, 0, (nPiezasTotales * altura) + altura);
19      TorresHanoi nPiezasTotales, pOrig, pBaseDestino, pBaseInterm, altura;
20  ENDPROC
21
22  PROC TorresHanoi (num nPiezas, robtarget pOrig, robtarget pDest, robtarget pInt, num altoPieza)
23
24      VAR robtarget pAproxOrigen;
25      VAR robtarget pAproxDestino;
26      VAR robtarget pAproxInter;
27      VAR robtarget pAux1;
28      VAR robtarget pAux2;
29
30      !Definimos los puntos de aproximación
31      pAproxOrigen := Offs (pOrig, 0, 0, altura * nPiezasTotales + altura);
32      pAproxDestino := Offs (pDest, 0, 0, altura * nPiezasTotales + altura);
33      pAproxInter := Offs (pInt, 0, 0, altura * nPiezasTotales + altura);
34
35      IF nPiezas = 1 THEN
36
37          !Movemos la pieza del origen al destino
38          MoveL pAproxOrigen, v2000, fine,Ventosa\WObj:=wobj0;
39          MoveLDO pOrig, v2000, fine,Ventosa\WObj:=wobj0, SD_ActivaVentosa, 1;

39          MoveLDO pOrig, v2000, fine,Ventosa\WObj:=wobj0, SD_ActivaVentosa, 1;
40          WaitDI ED_PiezaCogida, 1;
41
42          MoveL pAproxDestino, v2000, fine,Ventosa\WObj:=wobj0;
43          MoveLDO pDest, v2000, fine,Ventosa\WObj:=wobj0, SD_ActivaVentosa, 0;
44          WaitDI ED_PiezaCogida, 0;
45
46      ELSE
47
48          !Hacemos Hanoi(n-1, origen, destino, auxiliar)
49          pOrig := Offs (pOrig, 0, 0, -altura);
50          pDest := Offs (pDest, 0, 0, altura * nPiezas);
51          pInt := Offs (pInt, 0, 0, 0);
52          TorresHanoi nPiezas - 1, pOrig, pInt, pDest, altoPieza;
53
54          !Movemos la pieza del origen al destino
55          pOrig := Offs (pOrig, 0, 0, altura * nPiezas);
56          pDest := Offs (pDest, 0, 0, -altura);
57          pInt := Offs (pInt, 0, 0, 0);
58          MoveL pAproxOrigen, v2000, fine,Ventosa\WObj:=wobj0;
59          MoveLDO pOrig, v2000, fine,Ventosa\WObj:=wobj0, SD_ActivaVentosa, 1;
60          WaitDI ED_PiezaCogida, 1;
61
62          MoveL pAproxDestino, v2000, fine,Ventosa\WObj:=wobj0;
63          MoveLDO pDest, v2000, fine,Ventosa\WObj:=wobj0, SD_ActivaVentosa, 0;

64          WaitDI ED_PiezaCogida, 0;
65
66
67          !Hacemos Hanoi(n-1, auxiliar, origen, destino)
68          TorresHanoi nPiezas - 1, pInt, pDest, pOrig, altoPieza;
69
70      ENDIF
71
72  ENDPROC
```

Los valores de la altura no tienen mucho sentido, ya que intenté que los puntos que se le pasan como parámetro a la función al hacer la recursión tuvieran la altura de cada una de las torres, pero debido a la recursión no funcionó.

Simulaciones

Simulación 3 piezas





Como podemos observar, no se lleva a cabo la realización del ejercicio.

Activando y desactivando las señales de la ventosa, he podido comprobar que el algoritmo utilizado funciona correctamente, faltaría resolver el problema de las alturas para concluirlo.