**Brazo Robótico: Pruebas de Caja Blanca**

**Theta1.cs**

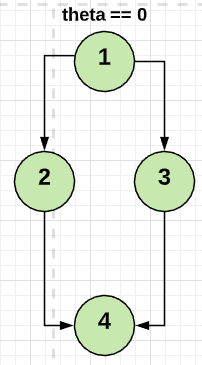
========== FUNCION Theta1.Update() ==========

1.SI theta = 0 ENTONCES

2.x <-- 0

SI NO ENTONCES

3.x <-- 360

4.FIN SI

V(G) = 2

Conjunto de Caminos:

* 1-2-4
* 1-3-4

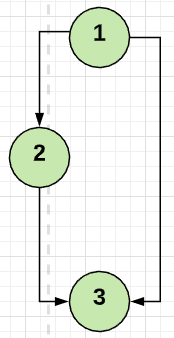
**Theta3.cs**

========== FUNCION Theta3.Update ==========

1.SI AngleDir(...) < 0 O DHParameters.getTheta2() > 90 ENTONCES

2.theta <-- theta \* -1

3.FIN SI



V(G) = 2

Conjunto de Caminos:

* 1-2-3
* 1-3

========== FUNCION Theta3.AngleDir ==========

1.SI dir > 0 ENTONCES

2.RETORNA 1

SI NO ENTONCES

3.SI dir < 0 ENTONCES

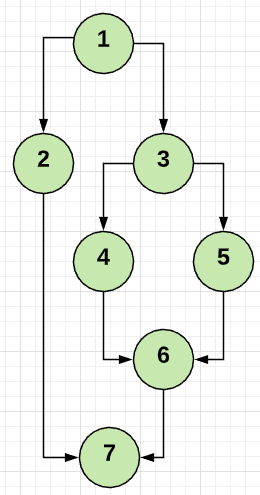
4.RETORNA -1

SI NO ENTONCES

5.RETORNA 0

6.FIN SI

7.FIN SI



V(G) = 2

Conjunto de Caminos:

* 1-2-7
* 1-3-4-6-7
* 1-3-5-6-7

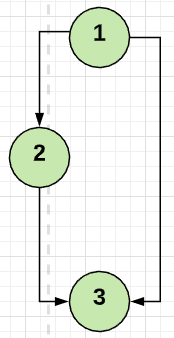
**UpperArmSlider.cs y LowerArmSlider.cs**

========== FUNCION UpperArmSlider.Start ==========

1.SI AngleDir(...) < 0 ENTONCES

2.theta\*=-1

3.FIN SI



V(G) = 2

Conjunto de Caminos:

* 1-2-3
* 1-3

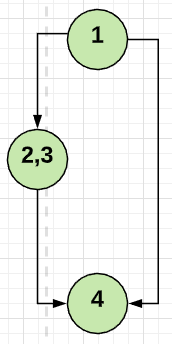
========== FUNCION UpperArmSlider.Update ==========

1.SI DHParameters.getMoveSlider() == TRUE ENTONCES

2.sliderTheta3.value <-- DHParameters.getTheta3()

3.lastTheta <-- DHParameters.getTheta3()

4.FIN SI



V(G) = 3

Conjunto de Caminos:

* 1-2-3-4
* 1-4

========== FUNCION UpperArmSlider.AngleDir ==========

1.SI dir > 0 ENTONCES

2.RETORNA 1

SI NO ENTONCES

3.SI dir < 0 ENTONCES

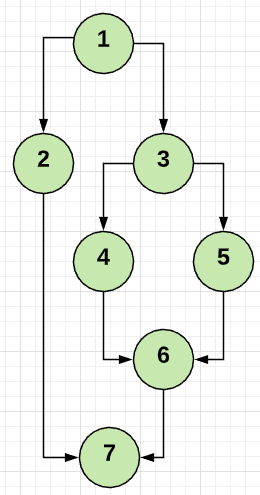
4.RETORNA -1

SI NO ENTONCES

5.RETORNA 0

6.FIN SI

7.FIN SI



V(G) = 2

Conjunto de Caminos:

* 1-2-7
* 1-3-4-6-7
* 1-3-5-6-7

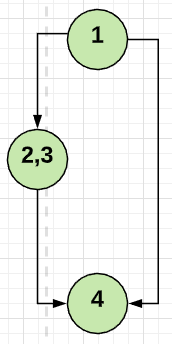
========== FUNCION UpperArmSlider.SliderJoint2 ==========

1.SI DHParameters.getMoveSlider() = false ENTONCES

2.RobotUpperArm.Rotate (-(angle - lastTheta), 0f , 0f)

3.lastTheta <-- angle

4.FIN SI



V(G) = 3

Conjunto de Caminos:

* 1-2-3-4
* 1-4

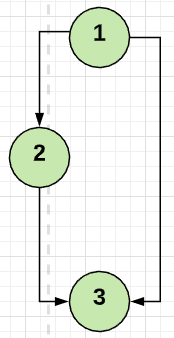
**KnotPoints.cs**

========== FUNCION KnotPoints.Start ==========

1.PARA i <-- 0 HASTA checks.Length CON PASO 1 HACER

2.checks[i].SetActive(false)

3.FIN PARA



V(G) = 2

Conjunto de Caminos:

* 1-2-3
* 1-3

========== FUNCION KnotPoints.SavePoint ==========

1.SI knotPoints < 5 ENTONCES

2.theta1Array[knotPoints] <-- BaseRotation.localRotation;

3.theta2Array[knotPoints] <-- Link1Rotation.localRotation;

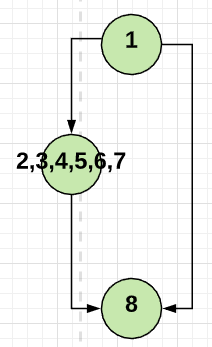
4.theta3Array[knotPoints] <-- Link2Rotation.localRotation;

5.positionsArm[knotPoints] <-- jointPos.transform.position;

6.checks[knotPoints].SetActive(true);

7.knotPoints++;

8.FIN SI



V(G) = 5

Conjunto de Caminos:

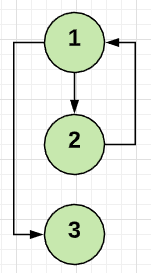
* 1-2-3-4-5-6-7-8
* 1-8

========== FUNCION KnotPoints.ClearPoints ==========

1.PARA i <-- 0 HASTA checks.Length CON PASO 1 HACER

2.checks[i].SetActive(false)

3.FIN PARA



V(G) = 2

Conjunto de Caminos:

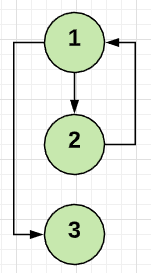
* 1-2-3
* 1-3

========== FUNCION KnotPoints.GoThroughPoints ==========

1.SI time.text != null ENTONCES

2.StartCoroutine(RotateMe(float.Parse(time.text)))

3.FIN SI



V(G) = 2

Conjunto de Caminos:

* 1-2-3
* 1-3

========== FUNCION KnotPoints.RotateMe ==========

1.PARA i <-- 0 HASTA knotPoints CON PASO 1 HACER

2.Quaternion baseFromAngle <-- BaseRotation.localRotation

3.Quaternion link1FromAngle <-- Link1Rotation.localRotation

4.Quaternion link2FromAngle <-- Link2Rotation.localRotation

5.PARA t <-- 0 HASTA < 1 CON PASO Time.deltaTime/inTime HACER

6.BaseRotation.localRotation <-- Quaternion.Lerp(...);

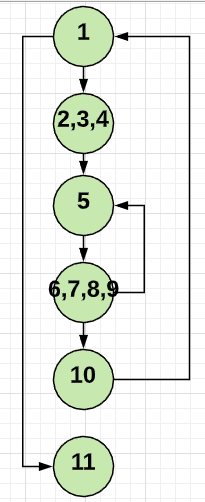
7.Link1Rotation.localRotation <-- Quaternion.Lerp(...);

8.Link2Rotation.localRotation <-- Quaternion.Lerp(...);

9.yield return null

10.FIN PARA

11.FIN PARA



V(G) = 5

Conjunto de Caminos:

* 1-2-3-4-5-6-7-8-9-10-11
* 1-11

========== FUNCION KnotPoints.DownloadCodeMELFA ==========

1.SI checks.Length > 0 ENTONCES

2....

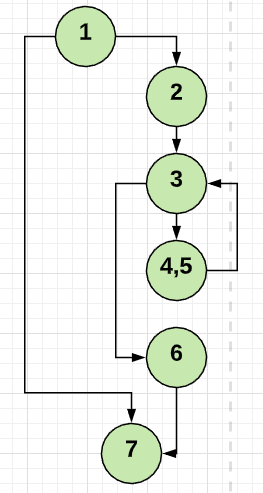
3.PARA i <-- 0 HASTA knotPoints CON PASO 1 HACER

4.positionAux <-- positionsArm[i];

5.sw.WriteLine(...);

6.FIN PARA

7.FIN SI



V(G) = 3

Conjunto de Caminos:

* 1-2-3-4-5-6-7
* 1-2-3-6-7
* 1-7