

INFORME PROJECTE I

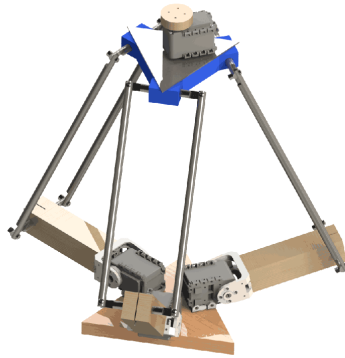
Construcció d'un robot delta per col·locar un circuit de dominó

Marc Asenjo i Ponce de León

Joan Marcè i Igual

Iñigo Moreno i Caireta

24 de juny de 2015



Índex

1	Objectiu	3
2	Informe mecànic	4
2.1	Assemblatge general	4
2.2	Servomotors	5
2.2.1	Plataforma suport	5
2.3	Braç - inicial	6
2.3.1	Estructura	6
2.3.2	Estabilitzador	6
2.4	Braç - final	7
2.5	Avantbraç	8
2.5.1	Vara unió	8
2.5.2	Eix Braç	8
2.5.3	Eix Pinça	9
2.5.4	Unió	9
2.6	Pinça	10
3	Informe cinemàtic	11
3.1	Cinemàtica inversa	11
3.1.1	Definicions de variables	11
3.1.2	Canvis de base	11
3.1.3	Càlcul d'un angle	12
3.2	Comprovació dels càlculs	13
3.3	Rang de treball	14
3.4	Implementació en Matlab	15
3.4.1	Càlcul d'un angle	15
3.4.2	Càlcul de tots els angles	16
3.4.3	Càlcul del rang de treball	17
4	Informe de programació	18
4.1	Control automàtic	18
4.1.1	Descripció del programa	18
4.1.2	Ús del programa	18
4.2	Unity	19
4.2.1	El programa	19
4.2.2	Documentació del codi	20

1 Objectiu

El nostre objectiu és construir un robot delta que sigui capaç de posicionar peces de dominó de la manera desitjada per l'usuari, aquest tindrà un programa que li permetrà dibuixar el circuit desitjat i també podrà controlar el robot mitjançant un joystick.

2 Informe mecànic

2.1 Assemblatge general

Aquest robot delta està format per quatre parts generals: els servos, els braços, els avantbraços i la pinça final.

El robot consta de tres servos que permetran el posicionament de la pinça; cada servomotor té acoblat un braç que es mou junt amb l'eix d'aquest. A més a més, els braços estan units mitjançant un eix amb l'avantbraç. Tots els avantbraços arriben a unir-se a la peça final que és la pinça permetent així que el moviment dels tres servos determini un punt a l'espai en el qual posicionar la pinça amb tres graus de llibertat de translació.

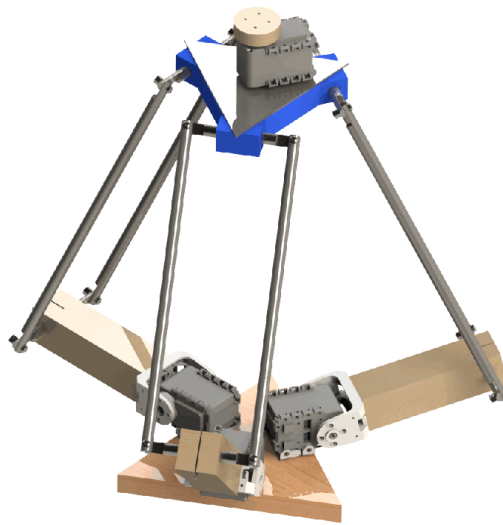


Figura 1: Assemblatge general

2.2 Servomotors

Els servomotors permeten el posicionament específic d'un eix a un cert angle. N'hi ha tres. S'utilitza el model AX-12 de dynamixel.

A part dels servos també s'han utilitzat els accessoris *FP04-F2* i *FP04-F3*. Per tal de facilitar l'ancoratge entre el servomotor i les diferents peces.

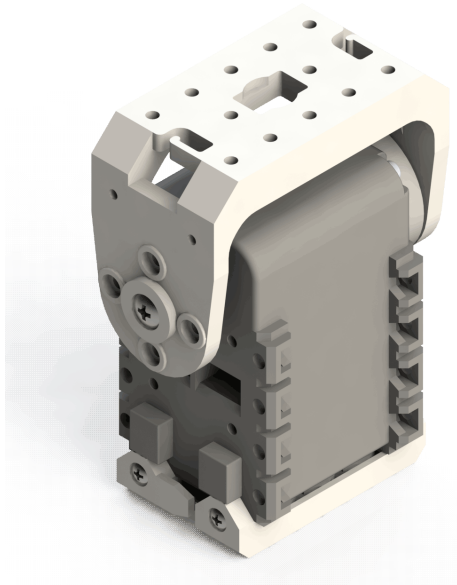


Figura 2: Imatge 3D del servomotor

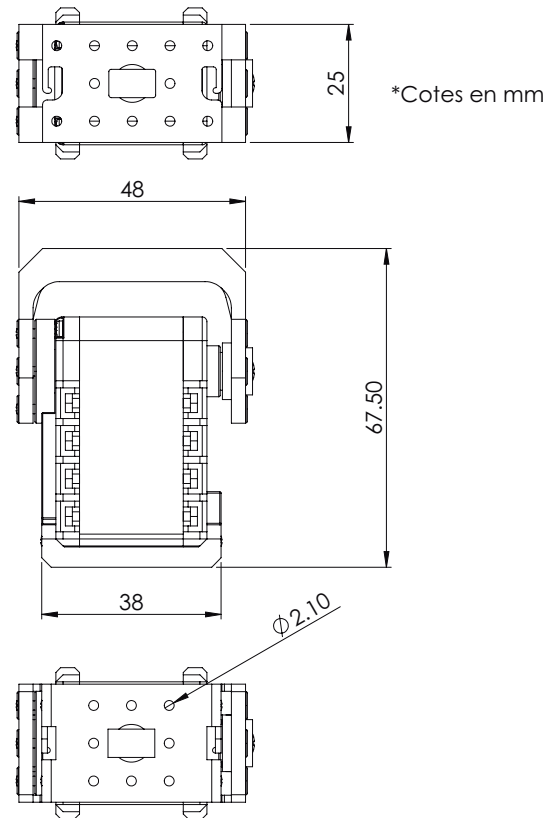


Figura 3: Plànol servo amb accessoris

2.2.1 Plataforma suport

Sobre aquesta plataforma estaran situats els tres servos del robot. Serà de fusta ja que permetrà cargolar els servomotors a sobre i així impedir un moviment no desitjat.

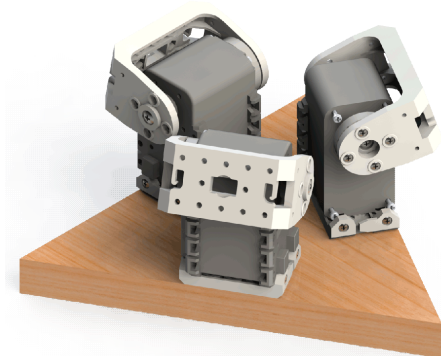


Figura 4: Plataforma amb els servos posats

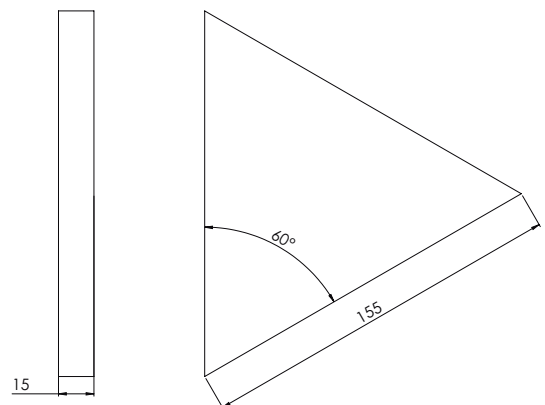


Figura 5: Plànol plataforma servos

2.3 Braç - inicial

Es recolza en els servomotors i principalment està fet de fusta. La part estructural és tota de fusta i està tota enganxada amb cola i després hi ha tres estabilitzadors que eviten vibracions innecessàries a la part del braç més propera a l'eix.



Figura 6: Braç muntat

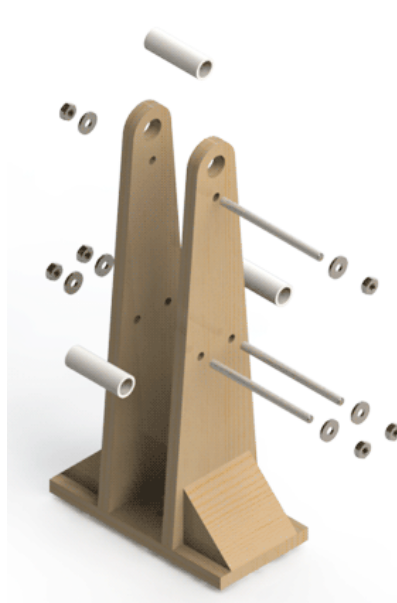


Figura 7: Muntatge del braç

2.3.1 Estructura

L'estructura està feta amb llistons de fusta. S'ha utilitzat un llistó de 45° d'inclinació per la part de suport que es troba entre la base i la columna vertical i per a fer tant la base com les columnes verticals s'ha utilitzat un llistó de 5 mm de gruix que s'ha tallat perquè tingui la forma desitjada.

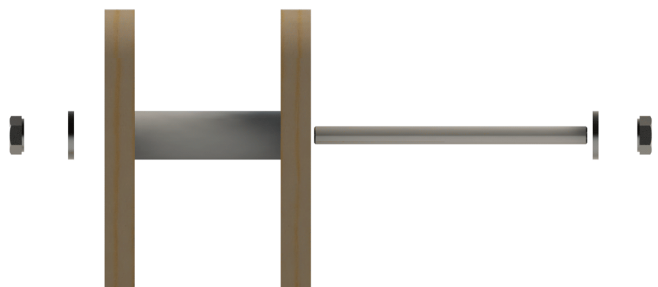
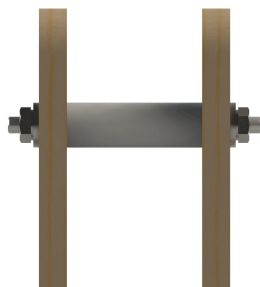
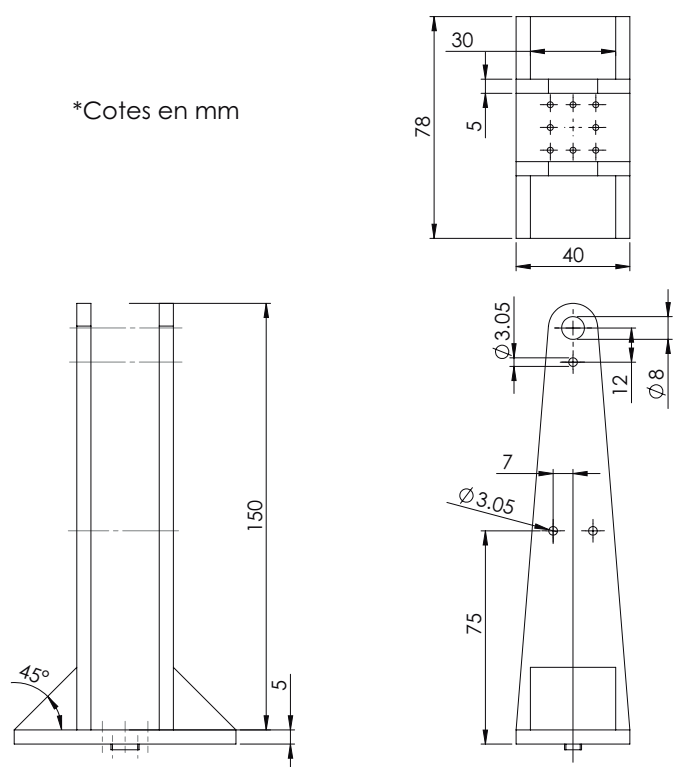
La raó per la que s'ha escollit fer-ho amb fusta és per poder fer-hi forats i talls amb relativa facilitat ja que la fusta és fàcil de treballar; també el reduït cost econòmic d'aquesta ha motivat escollir aquest material.

2.3.2 Estabilitzador

L'estabilitzador s'utilitza per tal que l'estructura de fusta no vibri ni es deformi degut al moviment del robot. És un conjunt de peces que està format per:

- Una barra roscada de M3 i 45 mm de llargada
- Un tub de plàstic de 6 mm de diàmetre interior i 8 mm de diàmetre exterior de 24 mm de llargada
- Dues volanderes
- Dues femelles M3

El tub de plàstic es situa entre les dues fustes verticals de l'estructura del braç i per dins hi va la barra roscada. A fora es col·loquen les femelles i les volanderes de manera que permeten prémer les dues fustes contra el tub de plàstic forçant així que la distància entre les dues fustes sigui constant.



2.4 Braç - final

El braç inicial era massa llarg així que es va haver de fer un nou model més curt, el nou model és més senzill ja que es va veure que un model massa elaborat era difícil de construir correctament i amb les mides especificades.



Figura 12: Nou braç

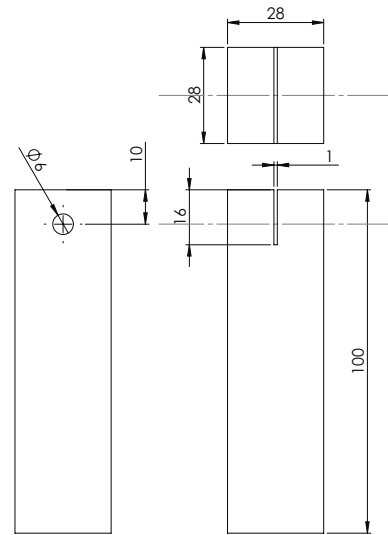


Figura 13: Nou braç

2.5 Avantbraç

L'avantbraç ha de connectar el braç amb la pinça i ha d'estar fet de tal manera que les dues rectes que hi ha entre els eixos sempre siguin paral·leles, a part la unió amb el braç i la pinça ha de ser mitjançant una junta esfèrica. En aquest robot s'ha separat la junta esfèrica en dos eixos per tal que faci la mateixa funció i sigui més fàcil de fabricar.



Figura 14: Avantbraç muntat



Figura 15: Muntatge avantbraç

2.5.1 Vara unió

És la que uneix l'eix situat a la pinça i l'eix situat al braç. És d'alumini que és un metall bastant lleuger i, a més a més, s'ha pensat en fer-la buida per dins per tal d'estalviar pes i així evitar errors en el posicionament final del robot.

2.5.2 Eix Braç

És el que es troba situat al braç de cada servo, el material utilitzat també ha estat l'alumini ja que és un material prou resistent com per fer d'eix i a la vegada també és fàcil de perforar.

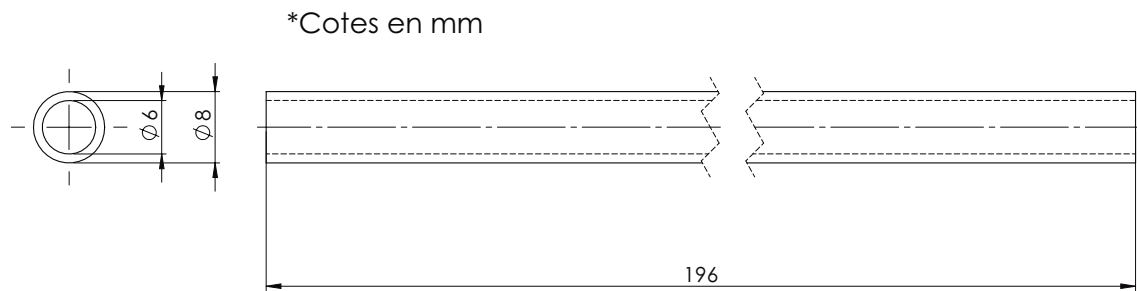


Figura 16: Plànol de la vara

Aquest eix té un petit forat just al mig per tal de poder posar-hi un passador i que així quedin units l'eix i un tub de plàstic que estarà situat entre les dues columnes del braç; d'aquesta manera s'espera que l'eix es mantingui centrat i lateralment. El tub de plàstic tindrà les mateixes mides que la separació entre les columnes.

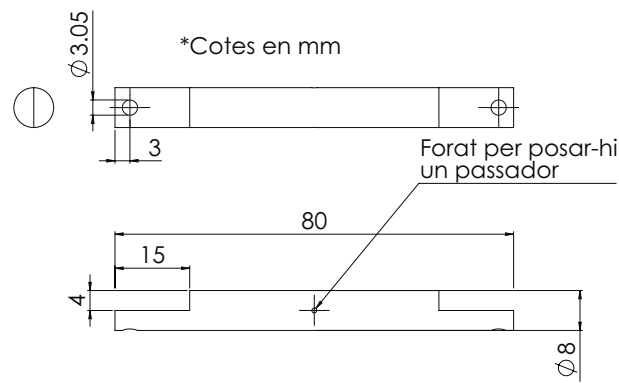


Figura 17: Plànol de l'eix

2.5.3 Eix Pinça

És l'eix que estarà situat al suport de la pinça, és exactament igual a l'eix del braç a excepció que aquest no estarà centrat mitjançant un passador. Un cop muntat al suport de la pinça, hi haurà una goma a cada costat de l'eix del suport de la pinça per evitar el moviment lateral.

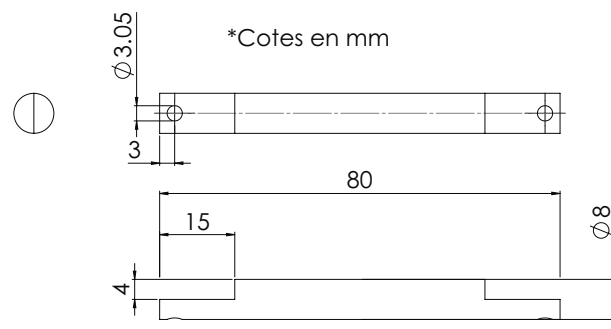
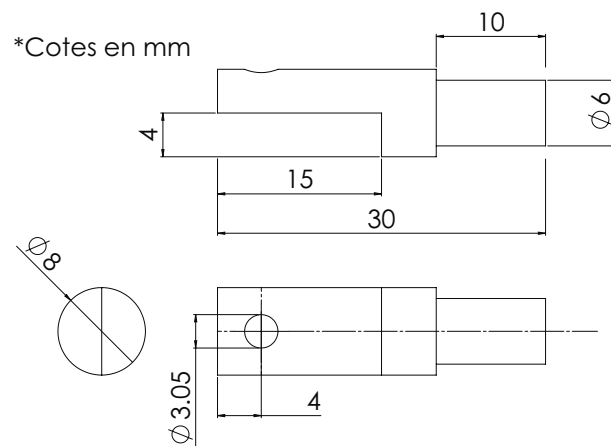


Figura 18: Plànol de l'eix de la pinça

2.5.4 Unió

Amb aquesta peça es pretén unir la vara a l'eix. Està feta de manera que la part que té un radi de 6 mm entri dins del forat interior de la vara i quedi fixat allà. D'aquesta manera s'acaba

tenint el mateix resultat a tenir una sola vara amb els forats d'unió a cada costat però amb un pes inferior.



2.6 Pinça

De moment només s'ha pensat el suport de la pinça ja que fins que no estigui aquesta part muntada i provat el seu correcte funcionament no es pot estar segur a seguir afegint més components. El suport serà de plàstic, imprès en 3D ja que la fusta aquí resulta ser un element massa dèbil i en el moment de fer els forats per on ha de passar l'eix, es trenca.

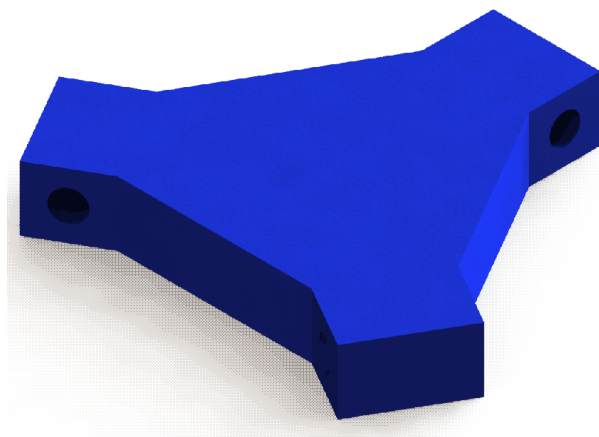


Figura 19: Representació 3D del suport de la pinça

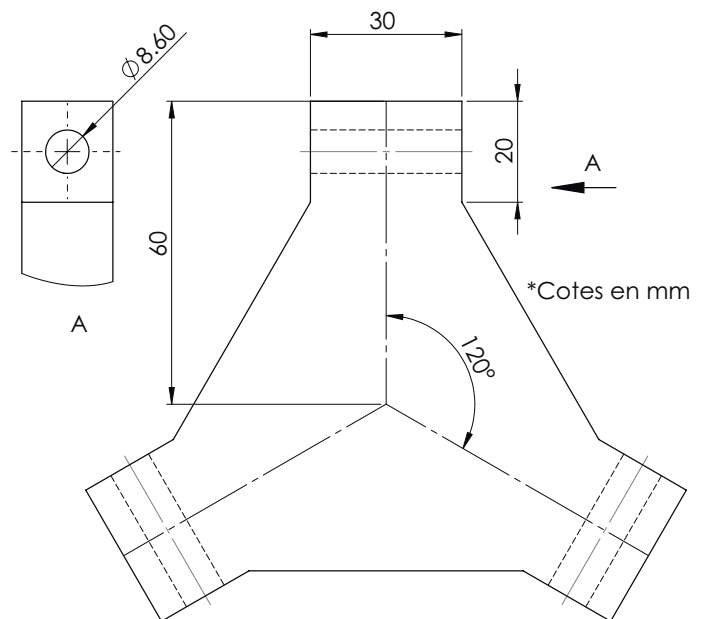


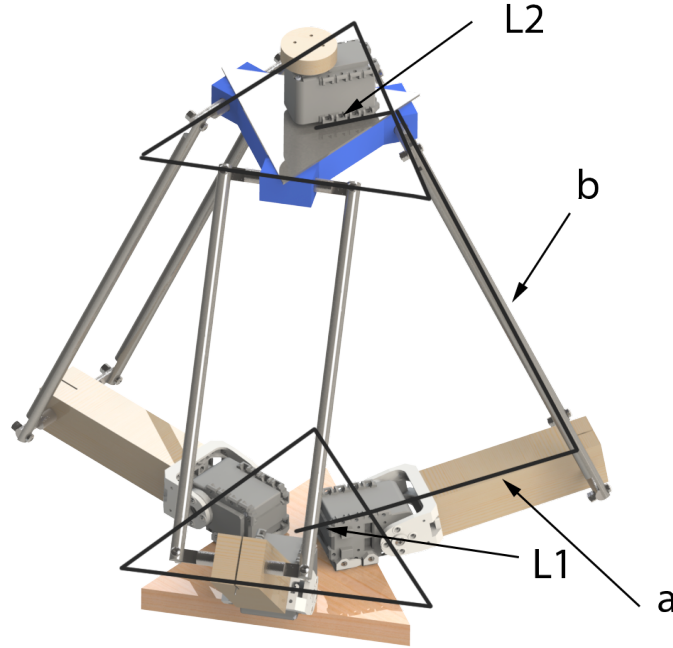
Figura 20: Plànol del suport de la pinça

3 Informe cinemàtic

3.1 Cinemàtica inversa

El càlcul invers és el que serveix per trobar els angles a partir de la posició a la que es vol situar la plataforma. Es pot fer el càlcul per cada motor de manera separada. Primer s'han de definir les mides principals del robot que s'usaran en el càlcul.

3.1.1 Definicions de variables



a és la mida total del braç, des de l'eix del motor fins a l'eix de la connexió amb l'avantbraç.

b és la mida de tot l'avantbraç des de la connexió amb el braç fins a la unió amb la plataforma.

L1 és la distància entre el centre de la base als eixos dels motors.

L2 és la distància entre el centre de la plataforma a l'eix de connexió amb l'avantbraç.

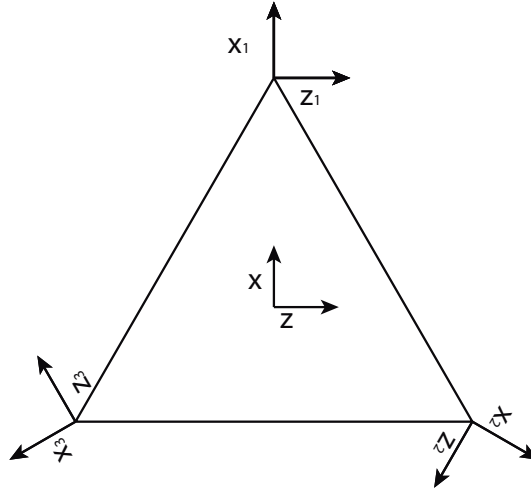
3.1.2 Canvis de base

Com el càlcul de cada angle és independent de la resta, es poden fer canvis de base per poder fer-ho tot amb una sola funció a l'hora de programar. La base inicial $\{x_0, y_0, z_0\}$ està al centre de la base, amb la x en la direcció del motor 1. Les bases $\{x_i, y_i, z_i\}$ que s'utilitzaran per calcular l'angle del motor i estan posicionades al centre del eix de cada motor, i la seva x apunta en la direcció perpendicular al eix del motor. També es suma $L2$ a x_i per fer que la posició objectiu sigui el punt de connexió entre la plataforma i l'avantbraç en lloc del centre de la plataforma.

$$\{x_1, y_1, z_1\} = \{x_0 + L2 - L1, y_0, z_0\}$$

$$\{x_2, y_2, z_2\} = \{z_0 \sin(60) - x_0 \cos(60) + L2 - L1, y_0, -z_0 \cos(60) - x_0 \sin(60)\}$$

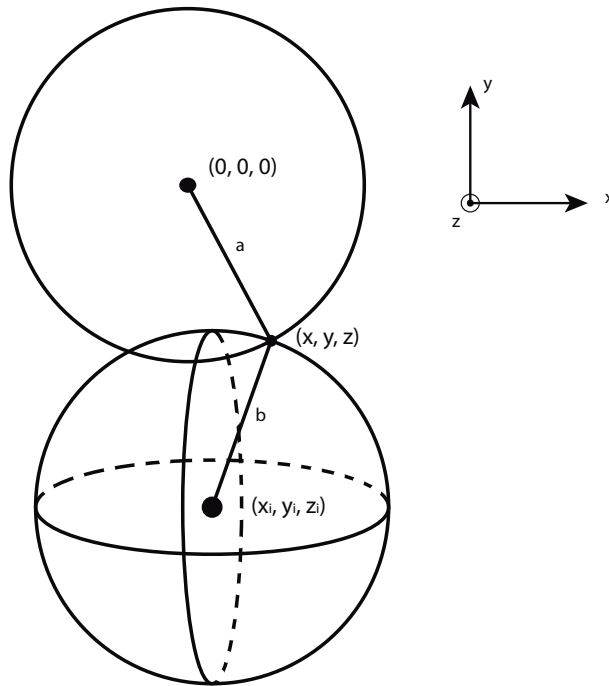
$$\{x_3, y_3, z_3\} = \{-z_0 \sin(60) - x_0 \cos(60) + L2 - L1, y_0, -z_0 \cos(60) + x_0 \sin(60)\}$$



3.1.3 Càlcul d'un angle

El motor només pot moure el final del braç en un cercle, gràcies a això es poden deduir dues formules:

$$x^2 + y^2 = a^2 \quad \text{i} \quad z = 0$$



També es pot veure que, com ha d'estar connectat a la plataforma amb una distancia b mitjançant l'avantbraç, s'ha de complir la formula:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = b^2$$

Ara només cal resoldre el sistema d'equacions.

$$x^2 - 2xx_i + x_i^2 + y^2 - 2yy_i + y_i^2 + z_i^2 = b^2$$

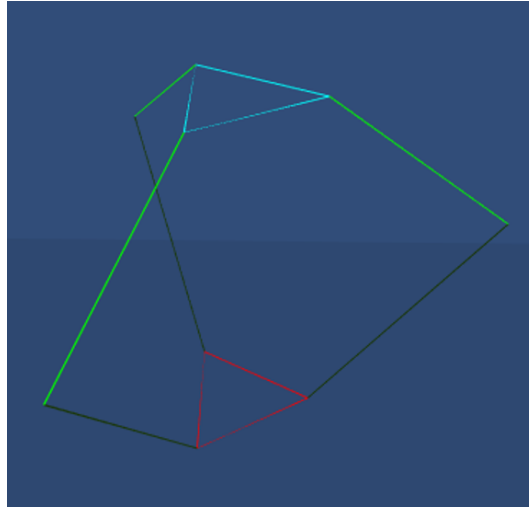
$$-2xx_i - 2yy_i = \underbrace{b^2 - a^2 - x_i^2 - y_i^2 - z_i^2}_n$$

$$-2x_i \sqrt{a^2 - y^2} = n + 2yy_i$$

$$\begin{aligned}
4a^2x_i^2 - 4y^2x_i^2 &= n^2 + 4yy_in + 4y^2y_i^2 \\
y^2(x_i^2 + y_i^2) + y(y_in) + \left(\frac{n^2}{4} - a^2x_i^2\right) &= 0 \\
y &= \frac{-y_i \pm \sqrt{y_i^2n^2 - 4(x_i^2 + y_i^2)\left(\frac{n^2}{4} - a^2x_i^2\right)}}{2(x_i^2 + y_i^2)} \\
x &= \pm \sqrt{a^2 - y^2} \\
\text{Finalment, } \theta &= \text{atan}\left(\frac{y}{x}\right)
\end{aligned}$$

3.2 Comprovació dels càlculs

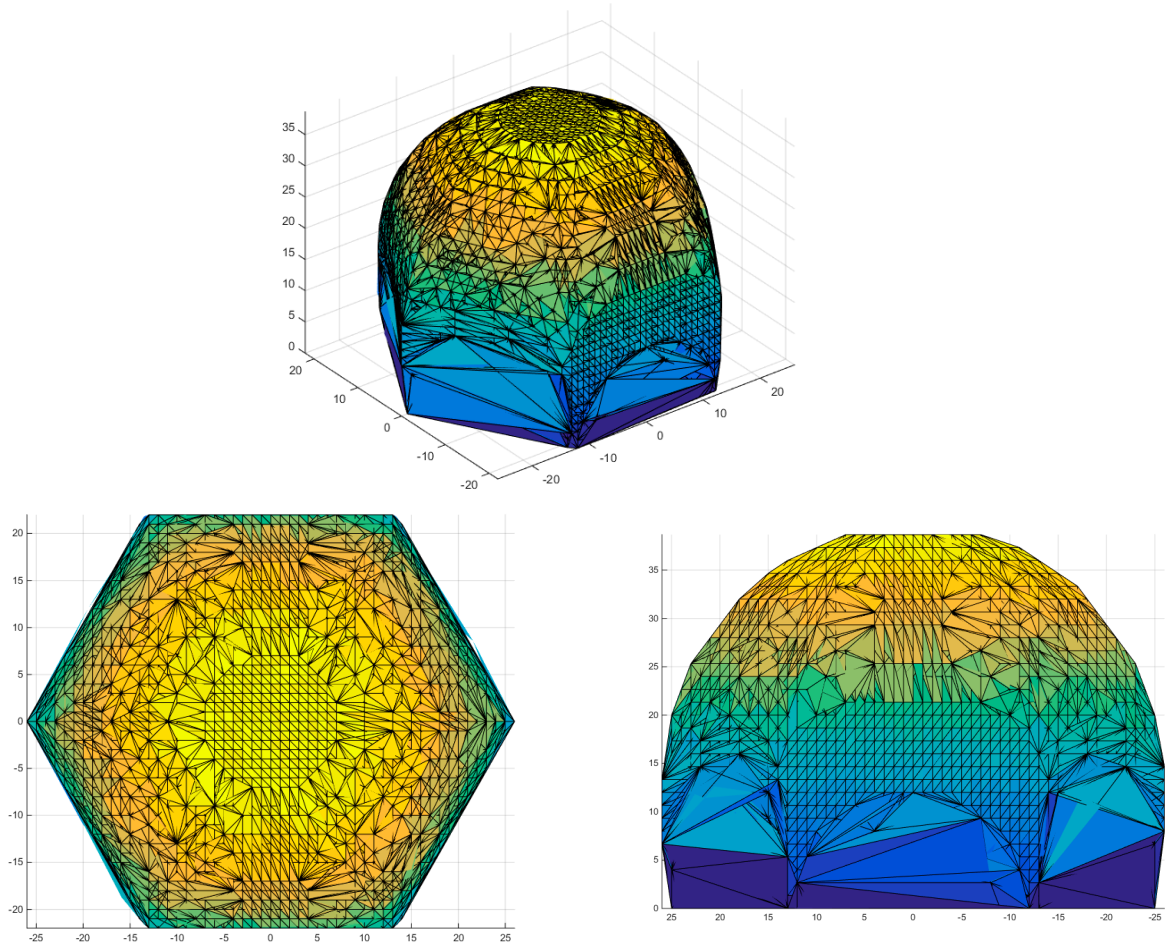
Per poder provar les nostres funcions sense trencar el robot, es va decidir fer una simulació en del Robot en Unity. La simulació és molt simple, Unity s'encarrega de dibuixar un esquema 3D del robot utilitzant línies a partir de la posició desitjada de la plataforma i els angles del robot calculats amb la funció trobada.



Aquesta aplicació també es va usar per determinar quin símbol s'ha de posar abans de les dos arrels. Per la primera arrel es va determinar que havia de anar acompanyada d'un signe negatiu només quan x_i és negatiu. La segona arrel només ha de ser negativa a partir de quan el ha d'estar completament cap a dalt. La condició exacta es: $b^2 - (y_i + a)^2 < x_i^2 + z_i^2$

3.3 Rang de treball

Per calcular el rang de treball s'ha utilitzat la fórmula de l'apartat anterior i buscant una gran quantitat de punts, d'aquests els que donaven un resultat possible s'han agafat com a vàlids i els altres s'han descartat. Tot seguit s'han passat els punts a Matlab® i s'ha generat la següent superfície.



Aquesta superfície està calculada amb tots els punts possibles teòrics però el robot té moltes limitacions mecàniques que no es tenen en compte en aquest càlcul. Per exemple els paral·lelograms dels avantbraços no poden tenir un angle qualsevol, ja que les barres de metall xoquen contra les plaques de fusta del braç si l'angle és molt petit.

3.4 Implementació en Matlab

La implementació de Matlab té tres parts, dues per al càlcul dels angles de treball i la tercera per poder calcular el rang de treball del robot.

3.4.1 Càlcul d'un angle

```
1  % Rep com a parmetres:
2  % - x, y, z de la posicio a calcular en la base adequada
3  % - a la llargada del brac
4  % - b la llargada de l'avantbrac
5  function angle = singleAngle(x0, y0, z0, a, b)
6
7  n = b^2 - a^2 - z0^2 - x0^2 - y0^2;
8  root = sqrt (n^2*y0^2 - 4*(x0^2 + y0^2)*(-x0^2*a^2 + n^2/4));
9
10 if (x0 < 0)
11     root = root * -1;
12 end
13
14 y = (-n*y0 + root ) / (2*(x0^2 + y0^2));
15
16 sign = 1;
17 if ((b^2 - (y0 + a)*(y0 + a)) < (x0^2 + z0^2) && x0 < 0)
18     sign = -1;
19 end
20
21 x = sqrt(a^2 - y^2)*sign;
22
23 if (isreal(x) && isreal(y))
24     angle = atan2(y, x);
25 else
26     angle = NaN;
27 end
28
29 end
```

3.4.2 Càlcul de tots els angles

```
1 % Rep com a parametre la posicio x, y, z on es vol saber els tres angles
2 % del robot que es el que retorna la funcio.
3 function D = setAngles(x, y, z)
4
5 % Mides del robot
6 %a = 17.233;      % Llargada brac
7 a = 12;
8 b = 22.648;      % Llargada avantbrac
9 L1 = 6.374;      % Distancia al centre del triangle de la base
10 L2 = 6;          % Distancia al centre del triangle de la pinca
11
12 % Definicions de constants
13 sin60 = sqrt(3)/2;
14 cos60 = 1/2;
15
16 % Calcul primer angle
17 x1 = x + L2 - L1;
18 y1 = -z;
19 z1 = y;
20 D(1) = singleAngle(x1,y1,z1, a, b);
21
22 % Calcul segon angle
23 x2 = y*sin60 - x*cos60 + L2 - L1;
24 y2 = -z;
25 z2 = -y*cos60 - x*sin60;
26 D(2) = singleAngle(x2,y2,z2, a, b);
27
28 % Calcul tercer angle
29 x3 = -y*sin60 - x*cos60 + L2 - L1;
30 y3 = -z;
31 z3 = -y*cos60 + x*sin60;
32 D(3) = singleAngle(x3,y3,z3, a, b);
33
34 for i = 1:3
35     D(i) = 150 - D(i)*180/pi;
36 end
37
38 end
```


3.4.3 Càlcul del rang de treball

```
1 % Es van provant tots els punts a l'espai definit per [-x, x], [-y, y] i
2 % [0, z] i si no retorna NaN (Not a Number) es considera com a valid i es
3 % guarda als vectors, tot seguit es genera una superfície en 3D junt amb el
4 % grafic.
5 function M = calcWorkspace(x, y, z, steps)
6 i = 1;
7 for dz = 0:z/(2*steps):z
8     for dx = -x:x/steps:x
9         for dy = -y:y/steps:y
10             D = setAngles(dx, dy, dz);
11             if (~isnan(D))
12                 bool = 1;
13                 for j=1:3
14                     if(D(j) > 250 || D(j) < 80)
15                         bool = 0;
16                     end
17                 end
18                 if(bool == 1)
19                     X(i) = dx;
20                     Y(i) = dy;
21                     Z(i) = dz;
22                     i = i + 1;
23                 end
24             end
25         end
26     end
27 end
28 end
29
30 tri = delaunay(X, Y, Z);
31 trisurf(tri,X, Y, Z, 'EdgeColor', 'black', 'LineWidth', 0.05);
32 axis('equal')
33
34 M(:, 1) = X;
35 M(:, 2) = Y;
36 M(:, 3) = Z;
37 end
```

4 Informe de programació

La programació esta dividida en dos programes, el de Control i el de Unity. El primer és el que fa "moure el robot" el segon és el que permet dibuixar el circuit que posteriorment serà col·locat. Tot el codi es pot trobar al següent repositori: <https://github.com/jmigual/deltaRobot>. Per descarregar els programes cal anar a l'apartat de **releases** i allà s'ha d'escollir la versió que es vulgui descarregar.

4.1 Control automàtic

4.1.1 Descripció del programa

Per fer el control automàtic s'ha utilitzat Qt 5.4 i també la API en C de Dynamixel de comunicació amb els servos. En aquest informe no es detalla molt aquesta part ja que a l'annex hi ha la documentació completa.

Al programa se li carrega un arxiu que conté totes les peces de dominó (extensió **.df**) que s'han de col·locar i executar el programa. Aquest llegeix l'arxiu i, mitjançant l'algoritme descrit a continuació, les va posant al seu lloc d'una en una. L'arxiu generalment l'haurà generat la part de Unity.

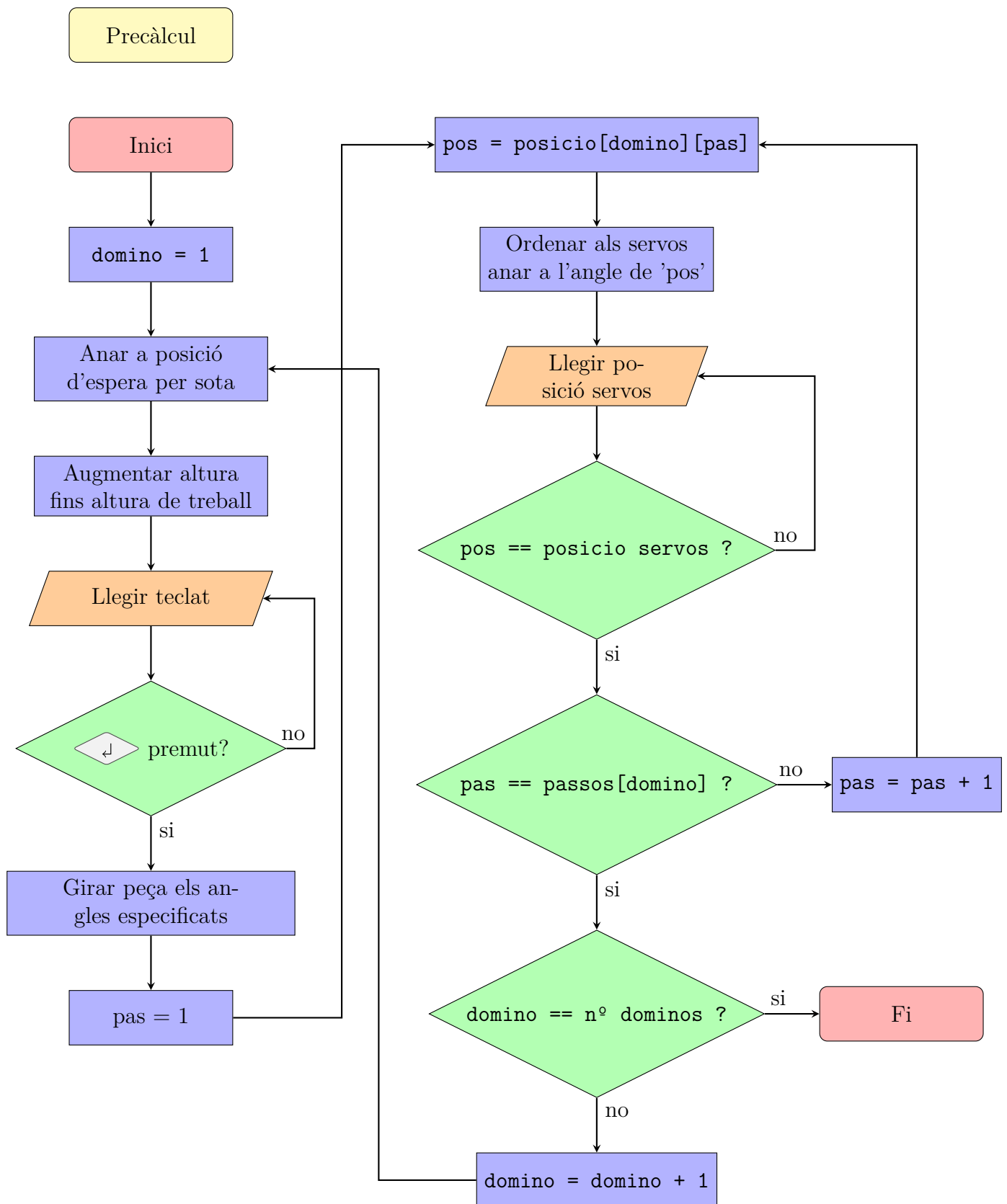
Per col·locar una peça es divideix la trajectòria en múltiples passos, aquí cada posició en un pas concret per a un dominó en concret es denota `posicio[domino][pas]`.

4.1.2 Ús del programa

Per fer anar el programa de control primer cal anar a **Edit** **Options** i allà seleccionar els ID dels servos que s'utilitzen per a cada braç, el braç que està a sobre la plataforma és el 1 i els altres dos són el 2 i el 3 en sentit antihorari. Un cop seleccionat els servos llavors es pot seleccionar el mode:

Manual permet controlar el robot mitjançant les tecles **W**, **A**, **S**, **D** per moure'l; **Q**, **E** per pujar i baixar i **H**, **J** per girar la punta del robot.

Automàtic començarà a executar el programa que se li hagi carregat prèviament, per carregar un programa anar a **File** **Import**. Cada vegada que se li posa una peça a la posició marcada s'ha de prémer **↵**.



4.2 Unity

4.2.1 El programa

En Unity s'ha volgut fer un editor complet en el que es pugui navegar per l'espai de treball, afegir peces amb diferents eines, provar si el circuit dissenyat és correcte i finalment enviar la posició i angle de gir de cada peça al programa que controla el robot. Funciona de la següent manera:

Un cop s'executa el programa es troba en el mode editor on el temps està aturat. En aquest mode es pot moure la càmera utilitzant els botons **W** **A** **S** **D** i canviar l'orientació fent clic i arrossegant el botó dret del ratolí. Si es fa clic central en un objecte la càmera automàticament es centrarà en ell.

En quant a editar, hi ha dues maneres de fer-ho. Es pot clicar el botó de baix a l'esquerra amb la imatge d'una peça de domino i automàticament apareixerà una peça al cursor que es pot moure lliurement pel pla del terra, i deixar-la tornant a clicar (botó esquerre del ratolí). Un cop fet això es pot seleccionar la peça, que es posarà de color vermell. Si es prem **T** apareixeran uns eixos que serveixen per acabar de col·locar de forma precisa la peça clicant i arrossegant l'eix que desitgem amb el clic esquerre. El mateix es pot fer per rotar la peça prement **R**, encara que en aquest cas només és útil una rotació, la de l'eix vertical. En qualsevol moment es pot cancel·lar el mode editor actual tornant a clicar la tecla amb què s'ha iniciat (**R** o **T**). Seguint exactament el mateix procediment però clicant el botó de **Add Force** es poden afegir unes fletxes que serviran de forces per comprovar que el circuit funciona.

La segona manera i la més còmode d'utilitzar per fer circuits de peces és la de "dibuixar". Si es clica **Let's draw!** la càmera automàticament es posarà en una vista aèria. Si ara es fa clic esquerre comença a posar peces per on es fa passar el ratolí, fins que es torna a fer clic. Les peces automàticament es posen en la direcció i distància les unes de les altres necessàries perquè el circuit funcioni, però pot ser que hagi sortit malament, així que si es prem **Del.** estant en el mode de dibuix totes les peces s'eliminen. Per acabar, es prem **Esc** i torna al mode editor, on es poden retocar les peces que es vulguin.

Per acabar, es fa clic a **Play**. La fletxa avançarà endavant fins trobar-se una peça de dominó, empenyent-la i començant la reacció en cadena del circuit. Es pot utilitzar la barra de dalt a la dreta per augmentar o disminuir la velocitat en la que cauen les peces, encara que no és recomanable pujar-la massa per qüestions de rendiment del programa. Si s'acaba satisfet amb el comportament del circuit, es pot tornar a clicar el botó que abans es deia **Play**, que ara té el nom de **Pause**. Les peces es tornaran a posar com estaven abans de ferles caure i llavors es pot acabar amb el botó **Save State**. Aquest botó actualitzara el document que s'utilitza per informar al programa principal de la posició de les peces.

4.2.2 Documentació del codi

En Unity s'han fet dos escenes, la principal (Main) i la del simulador del robot (**SimuladorRobot**). Els scripts que s'han usat al Main són:

- **CameraController**: Permet moure la càmera i fer zoom.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class CameraController : MonoBehaviour {
5
6     public GameObject center;
7     public bool centered = true;
8     public float distance=10f;
9     public float distanceMin=3f;
10    public float distanceMax=30f;
11    public float zoomspeed=500f;
12    public float theta;
```

```

13 public float phi;
14 public float cameraspeed=2000f;
15 public float camRayLength=100f;
16 public float objectiveSwitchSpeed=10f;
17 public float dragSpeed=10f;
18 public float movementSpeed=20f;
19 public float size = 5f;
20
21 public static bool onlyZoom = false;
22
23
24 private Transform ObjectiveInitial;
25 private Vector3 obj_pos;
26 private const float rad=2*Mathf.PI/360;
27 private Vector3 ObjectiveDesired;
28 private Vector2 LastMouse;
29 private bool switchingobjective=false;
30
31 private float desiredDistance;
32
33 void Start () {
34     desiredDistance=distance;
35     ObjectiveInitial=center.transform;
36     obj_pos = ObjectiveInitial.position;
37     ObjectiveDesired=obj_pos;
38 }
39
40 void Update () {
41     if (!onlyZoom) update_objective ();
42     update_angles ();
43     if (!onlyZoom) update_transform();
44 }
45
46
47 void update_objective (){
48     if (Input.GetKeyDown (KeyCode.Mouse2)){
49         Ray CamRay = Camera.main.ScreenPointToRay(Input.mousePosition);
50         RaycastHit floorHit;
51         if (Physics.Raycast (CamRay, out floorHit, camRayLength, 1)) {
52             switchingobjective=true;
53             centered=true;
54             center=floorHit.collider.gameObject;
55         }
56     }
57     if (Input.GetKey(KeyCode.A)){
58         if (centered){
59             centered=false;
60             switchingobjective=false;
61             ObjectiveDesired=obj_pos;
62         }
63         Vector3 desp=new Vector3 (Mathf.Sin(theta*rad),0,-Mathf.Cos(theta*rad));
64         obj_pos+=desp*movementSpeed*Time.fixedDeltaTime;
65     }
66     if (Input.GetKey(KeyCode.D)){
67         if (centered){
68             centered=false;
69             switchingobjective=false;
70             ObjectiveDesired=obj_pos;
71         }
72         Vector3 desp=new Vector3 (-Mathf.Sin(theta*rad),0,Mathf.Cos(theta*rad));
73         obj_pos+=desp*movementSpeed*Time.fixedDeltaTime;
74     }
75     if (Input.GetKey(KeyCode.W)){
76         if (centered){
77             centered=false;
78             switchingobjective=false;
79             ObjectiveDesired=obj_pos;
80         }
81         Vector3 desp=new Vector3 (-Mathf.Cos(theta*rad),0,-Mathf.Sin(theta*rad));
82         obj_pos+=desp*movementSpeed*Time.fixedDeltaTime;
83     }
84     if (Input.GetKey(KeyCode.S)){
85         if (centered){
86             centered=false;
87             switchingobjective=false;
88             ObjectiveDesired=obj_pos;

```

```

89     }
90     Vector3 desp=new Vector3 (Mathf.Cos(theta*rad),0,Mathf.Sin(theta*rad));
91     obj_pos+=desp*movementSpeed*Time.fixedDeltaTime;
92 }
93 if (centered)ObjectiveDesired=center.transform.position;
94 if (switchingobjective){
95     Vector3 error=ObjectiveDesired-obj_pos;
96     obj_pos=Vector3.Lerp(obj_pos, ObjectiveDesired, objectiveSwitchSpeed*Time.fixedDeltaTime
97         /(Mathf.Pow (error.magnitude,0.5f)));
98     if (error.magnitude<0.02)switchingobjective=false;
99 }
100 else{
101     if (centered)obj_pos=ObjectiveDesired;
102 }
103 void update_angles (){
104     desiredDistance = Mathf.Clamp(desiredDistance - Input.GetAxis("Mouse ScrollWheel")*
105         zoomspeed*Time.fixedDeltaTime, distanceMin, distanceMax);
106     distance=Mathf.Lerp (distance,desiredDistance,10*Time.fixedDeltaTime);
107     size -= zoomspeed*Time.fixedDeltaTime*Input.GetAxis("Mouse ScrollWheel");
108     size = Mathf.Clamp(size,1f,100f);
109     Camera.main.orthographicSize=size;
110     if (!onlyZoom) {
111         if (Input.GetKeyDown (KeyCode.Mouse1))
112             LastMouse = Input.mousePosition;
113         if (Input.GetKey (KeyCode.Mouse1)) {
114             Vector2 dist = Input.mousePosition;
115             dist -= LastMouse;
116             theta -= dist.x * dragSpeed;
117             phi += dist.y * dragSpeed;
118             LastMouse = Input.mousePosition;
119         }
120         theta = theta % 360;
121         phi = Mathf.Clamp (phi, 0.05f, 179.95f);
122     }
123 }
124 void update_transform (){
125     Vector3 despl = new Vector3(Mathf.Sin(phi*rad)*Mathf.Cos(theta*rad),
126         Mathf.Cos(phi*rad),
127         Mathf.Sin(phi*rad)*Mathf.Sin(theta*rad));
128     despl*=distance;
129     transform.position = obj_pos + despl;
130     transform.rotation = Quaternion.LookRotation (-despl);
131 }
132
133 }

```

- dominoPosition: Mou les peces de domino mentre les estas col·locant.

```

1 using UnityEngine;
2 using System.Collections;
3
4 public class dominoPosition : MonoBehaviour {
5     public float height;
6     public bool forceApplier;
7     public bool adding = false;
8     public float speed = 10f;
9
10    private bool shot = false;
11
12    void Start () {
13    }
14
15    void Update () {
16        if (adding) {
17            Ray camRay = Camera.main.ScreenPointToRay (Input.mousePosition);
18            RaycastHit floorHit;
19
20            if (Physics.Raycast (camRay, out floorHit, 100f, LayerMask.GetMask ("Floor"))) {
21                transform.position = floorHit.point + height*Vector3.up;
22            }
23        }
24
25        if (Input.GetKeyDown (KeyCode.Mouse0)){
26            adding = false;

```

```

27
28     }
29
30     if (forceApplier && !shot && Time.timeScale==Editor.timescale) {
31         GetComponent<Rigidbody>().velocity = transform.forward * speed;
32     }
33 }
34
35 void OnCollisionEnter (Collision collision){
36     if (forceApplier && collision.gameObject.tag == "Player") {
37         gameObject.SetActive(false);
38     }
39 }
40 }

```

- Draw: Afegeix tota la funcionalitat de dibuixar circuits de domino.

```

1 using UnityEngine;
2 using System.Collections;
3
4 public class Draw : MonoBehaviour {
5     public GameObject canvas;
6     public GameObject dominoPiece;
7     public float distance = 1.5f;
8
9     private bool drawing = false;
10    private Vector3 lastPosition;
11    private bool first = true;
12    private GameObject lastDomino;
13
14    void Start () {
15        canvas.SetActive (false);
16    }
17
18    void Update () {
19        canvas.SetActive (false);
20        if (Input.GetKeyDown (KeyCode.Mouse0))
21        {
22            drawing = !drawing;
23            first = true;
24            lastPosition = new Vector3(100,100,100);
25        }
26
27        if (Input.GetKeyDown (KeyCode.Escape) || Input.GetKeyDown (KeyCode.Return))
28        {
29            CameraController.onlyZoom = false;
30            Camera.main.orthographic = false;
31            GetComponent<Editor> ().enabled = true;
32
33            canvas.SetActive (true);
34
35            GetComponent<Draw> ().enabled = false;
36        }
37
38        if (Input.GetKeyDown (KeyCode.Delete))
39        {
40            GameObject[] pieces = GameObject.FindGameObjectsWithTag ("Player");
41            for (int i = 0; i<pieces.Length; ++i) Destroy (pieces[i]);
42        }
43
44        if (drawing)
45        {
46            Ray camRay = Camera.main.ScreenPointToRay (Input.mousePosition);
47            RaycastHit floorHit;
48
49            if (Physics.Raycast (camRay, out floorHit, 1000f, LayerMask.GetMask ("Floor")))
50            {
51                if (Vector3.Distance(floorHit.point, lastPosition)>= distance)
52                {
53                    if (!first) {
54                        lastDomino.transform.rotation = Quaternion.LookRotation(floorHit.point -
55                            lastPosition);
56                        lastDomino = Instantiate (dominoPiece, lastPosition+(floorHit.point-lastPosition).
57                            normalized*distance + 1.2f*dominoPiece.transform.localScale.y*Vector3.up,
58                            Quaternion.LookRotation(floorHit.point - lastPosition)) as GameObject;
59                        lastPosition = lastPosition+(floorHit.point-lastPosition).normalized*distance;

```

```

57         }
58         else {
59             first = false;
60             lastDomino = Instantiate (dominoPiece, floorHit.point + 1.2f*dominoPiece.transform
                                   .localScale.y*Vector3.up, Quaternion.LookRotation(Vector3.forward)) as
                                   GameObject;
61             lastPosition=floorHit.point;
62         }
63     }
64 }
65 }
66 }
67 }

```

- Editor: Permet moure i rotar els objectes ja col·locats.

```

1  using UnityEngine;
2  using System.Collections;
3  using UnityEngine.UI;
4  using System.IO;
5  using System;
6
7  public class Editor : MonoBehaviour {
8
9      public Color selectedColor = Color.red;
10     public GameObject X,Y,Z;
11     public GameObject RX, RY, RZ;
12     public float arrowlength = 10f;
13     public float translateSpeed = 25f;
14     public string mode = "t";
15     public bool relativeRotation = true;
16     public GameObject DominoPiece;
17     public GameObject Force;
18     public GameObject Stair;
19     public string path;
20     public Slider slider;
21     static public float timescale = 2f;
22
23     private GameObject selected;
24     private bool selectedExists;
25     private float camRayLength;
26     private Color lastColor;
27     private bool dragging=false;
28     private GameObject draggingAxis,draggingRotator;
29     Vector2 LastMouse;
30     Vector3 toLastMouseR;
31     private string lastMode;
32     private Quaternion RXinitial, RYinitial, RZinitial;
33
34     void Start () {
35         Physics.gravity = 19.6f * Vector3.down;
36         Time.timeScale=0f;
37         selected = Camera.main.GetComponent<CameraController>().center;
38         selectedExists = false;
39         camRayLength = GameObject.FindGameObjectWithTag("MainCamera").GetComponent<
                                   CameraController>().camRayLength;
40         if (relativeRotation){
41             RXinitial=RX.transform.rotation;
42             RYinitial=RY.transform.rotation;
43             RZinitial=RZ.transform.rotation;
44         }
45         X.SetActive (true); Y.SetActive (true); Z.SetActive (true); RX.SetActive (true); RY.
                                   SetActive (true); RZ.SetActive (true);
46         X.GetComponentInChildren<Renderer>().material.color = Color.red;
47         Y.GetComponentInChildren<Renderer>().material.color = Color.green;
48         Z.GetComponentInChildren<Renderer>().material.color = Color.blue;
49         RX.GetComponentInChildren<Renderer>().material.color = Color.red;
50         RY.GetComponentInChildren<Renderer>().material.color = Color.green;
51         RZ.GetComponentInChildren<Renderer>().material.color = Color.blue;
52     }
53
54
55     void Update () {
56         align_tools ();
57         update_mode ();
58         update_selected();

```



```

59     align_tools ();
60 }
61
62 void align_tools (){
63     X.SetActive(mode=="t" & selectedExists);
64     Y.SetActive(mode=="t" & selectedExists);
65     Z.SetActive(mode=="t" & selectedExists);
66     RX.SetActive(mode=="r" & selectedExists);
67     RY.SetActive(mode=="r" & selectedExists);
68     RZ.SetActive(mode=="r" & selectedExists);
69     if (selectedExists) {
70         X.transform.position = selected.transform.position;
71         Y.transform.position = selected.transform.position;
72         Z.transform.position = selected.transform.position;
73         RX.transform.position = selected.transform.position;
74         RY.transform.position = selected.transform.position;
75         RZ.transform.position = selected.transform.position;
76         if (relativeRotation){
77             RX.transform.rotation = selected.transform.rotation*RXinitial;
78             RY.transform.rotation = selected.transform.rotation*RYinitial;
79             RZ.transform.rotation = selected.transform.rotation*RZinitial;
80         }
81     }
82 }
83
84 void update_mode (){
85     if (Time.timeScale==0){
86         if (Input.GetKeyDown(KeyCode.T)){
87             if (mode=="t")mode="-";
88             else if (selectedExists) mode="t";
89         }
90         if (Input.GetKeyDown(KeyCode.R)){
91             if (mode=="r")mode="-";
92             else if (selectedExists) mode="r";
93         }
94     }
95 }
96
97 void update_selected (){
98     if (Input.GetKeyDown (KeyCode.Delete)) {
99         Destroy (selected);
100         selectedExists = false;
101         mode = "-";
102         Camera.main.GetComponent <CameraController>().centered = false;
103     }
104     if (Input.GetKeyDown (KeyCode.Mouse0)){
105         Ray CamRay = Camera.main.ScreenPointToRay(Input.mousePosition);
106         RaycastHit floorHit;
107         if (Physics.Raycast (CamRay, out floorHit, camRayLength, 1)) {
108             if (floorHit.collider.gameObject.CompareTag("Axis")){
109                 LastMouse = Input.mousePosition;
110                 dragging = true;
111                 char auxc = floorHit.collider.gameObject.name[0];
112                 if (auxc == 'X')draggingAxis = X;
113                 if (auxc == 'Y')draggingAxis = Y;
114                 if (auxc == 'Z')draggingAxis = Z;
115             }
116             else if (floorHit.collider.gameObject.CompareTag("Rotator")){
117                 dragging = true;
118                 char auxc = floorHit.collider.gameObject.name[0];
119                 if (auxc == 'X')draggingRotator = RX;
120                 if (auxc == 'Y')draggingRotator = RY;
121                 if (auxc == 'Z')draggingRotator = RZ;
122                 Ray MouseRay = Camera.main.ScreenPointToRay(Input.mousePosition);
123                 float t=Vector3.Dot (draggingRotator.transform.position,draggingRotator.transform.
                    forward)-Vector3.Dot (draggingRotator.transform.forward,MouseRay.origin);
124                 t=t/Vector3.Dot (draggingRotator.transform.forward,MouseRay.direction);
125                 toLastMouseR=MouseRay.origin+t*MouseRay.direction-draggingRotator.transform.position
                    ;
126             }
127             else{
128                 if (selectedExists) selected.GetComponentInChildren<Renderer>().material.color =
                    lastColor;
129                 if (floorHit.collider.gameObject.CompareTag("EditorOnly")) selected = floorHit.
                    collider.transform.parent.gameObject;
130                 else selected = floorHit.collider.gameObject;

```

```

131         lastColor = selected.GetComponentInChildren<Renderer>().material.color;
132         Renderer[] letsColor = selected.GetComponentsInChildren<Renderer>();
133         for (int i=0; i<letsColor.Length; ++i) letsColor[i].material.color = selectedColor;
134         selectedExists = true;
135     }
136 }
137 }
138 if (Input.GetKeyUp(KeyCode.Mouse0))dragging = false;
139
140 if (Input.GetKey (KeyCode.Mouse0) && dragging && mode=="t"){
141     Vector2 despMouse = Input.mousePosition;
142     despMouse -= LastMouse;
143     LastMouse=Input.mousePosition;
144     Vector2 axisScreenTip = Camera.main.WorldToScreenPoint(draggingAxis.transform.position+
        draggingAxis.transform.up*arrowlength);
145     Vector2 axisScreenBase = Camera.main.WorldToScreenPoint(draggingAxis.transform.position)
        ;
146     Vector2 axisScreen = axisScreenTip-axisScreenBase;
147     float res = Vector2.Dot (despMouse,axisScreen)/Mathf.Pow(axisScreen.magnitude,2);
148     selected.transform.position+=draggingAxis.transform.up*res*translateSpeed;
149 }
150 if (Input.GetKey (KeyCode.Mouse0) && dragging && mode=="r"){
151     Ray MouseRay = Camera.main.ScreenPointToRay(Input.mousePosition);
152     float t=Vector3.Dot (draggingRotator.transform.position,draggingRotator.transform.
        forward)-Vector3.Dot (draggingRotator.transform.forward,MouseRay.origin);
153     t=t/Vector3.Dot (draggingRotator.transform.forward,MouseRay.direction);
154     Vector3 toPos=MouseRay.origin+t*MouseRay.direction-draggingRotator.transform.position;
155     float dAngle = Vector3.Angle(toLastMouseR , toPos);
156     if (Vector3.Dot (draggingRotator.transform.forward,Vector3.Cross (toLastMouseR,toPos))
        <0)dAngle=-dAngle;
157     toLastMouseR=toPos;
158     selected.transform.Rotate(draggingRotator.transform.forward*dAngle,Space.World);
159 }
160 }
161
162 public void PausePlay (){
163     if (Time.timeScale == timescale) {
164         Time.timeScale = 0;
165         GameObject.FindGameObjectWithTag ("Pause").GetComponentInChildren <Text> ().text =
            "Play";
166         mode = lastMode;
167         loadState();
168     }
169     else {
170         if (selectedExists) selected.GetComponentInChildren<Renderer>().material.color =
            lastColor;
171         selectedExists = false;
172         Time.timeScale = timescale;
173         GameObject.FindGameObjectWithTag("Pause").GetComponentInChildren <Text>().text = "
            Pause";
174         lastMode = mode;
175         mode = "-";
176
177         string aux=path;
178         path="./temp.df";
179         saveState();
180         path=aux;
181     }
182 }
183
184 public void AddD (){
185     if (Time.timeScale == 0) {
186         GameObject piece = (GameObject)Instantiate (DominoPiece, DominoPiece.transform.
            position, DominoPiece.transform.rotation);
187         piece.GetComponent<dominoPosition> ().adding = true;
188     }
189 }
190 }
191
192 public void AddF (){
193     if (Time.timeScale == 0) {
194         GameObject piece = (GameObject)Instantiate (Force, Force.transform.position, Force
            .transform.rotation);
195         piece.GetComponent<dominoPosition> ().adding = true;
196         selected=piece;
197     }

```

```

198     }
199
200     public void AddS (){
201         if (Time.timeScale == 0) {
202             GameObject piece = (GameObject)Instantiate (Stair, Stair.transform.position, Stair.
                transform.rotation);
203             piece.GetComponent<dominoPosition> ().adding = true;
204         }
205     }
206
207     public void LetsDraw (){
208         if (selectedExists) selected.GetComponentInChildren<Renderer>().material.color = lastColor
            ;
209         selectedExists = false;
210         mode = "-";
211         GetComponent<Draw> ().enabled = true;
212         CameraController.onlyZoom = true;
213         Camera.main.orthographic = true;
214         Camera.main.transform.position = new Vector3 (0, 100, 0);
215         Camera.main.transform.rotation = Quaternion.LookRotation (Vector3.down);
216         GetComponent<Editor> ().enabled = false;
217     }
218
219     public void TimeScale (){
220         timescale = slider.value;
221         Time.timeScale = slider.value;
222     }
223
224     public void saveState (){
225         GameObject[] pieces = GameObject.FindGameObjectsWithTag ("Player");
226         string content = pieces.Length.ToString() + Environment.NewLine;
227         for (int i = 0; i<pieces.Length; ++i)
228             content = content + pieces [i].transform.position.x.ToString("F2") + "□" +
229                 pieces [i].transform.position.z.ToString("F2") + "□" +
230                 pieces[i].transform.rotation.eulerAngles.y.ToString("F2") + Environment.
                NewLine;
231         System.IO.File.WriteAllText(path, content);
232     }
233
234     void loadState (){
235         GameObject[] pieces = GameObject.FindGameObjectsWithTag("Player");
236         foreach (GameObject piece in pieces) Destroy (piece,0.0f);
237         selectedExists=false;
238         StreamReader sr = new StreamReader("./temp.df");
239         string fileContents = sr.ReadToEnd();
240         sr.Close();
241
242         string[] lines = fileContents.Split(new char[] {'\n'});
243         for (int i=1; i<lines.Length-1; i++) {
244             string[] vars = lines[i].Split(new char[] { ' ' });
245             Vector3 newpos= new Vector3 (float.Parse (vars[0]),0.5f,float.Parse(vars[1]));
246             Instantiate (DominoPiece, newpos+ 1.2f*DominoPiece.transform.localScale.y*Vector3.up,
                Quaternion.Euler(0,float.Parse(vars[2]),0));
247         }
248     }
249 }

```

Els scripts que s'han usat a SimuladorRobot són:

- CameraController: (el mateix que a Main).
- RobotCreator: Dibuixa el robot en 3D a partir dels paràmetres públics que se li passin.

```

1 using UnityEngine;
2 using System.Collections;
3
4 public class RobotCreator : MonoBehaviour {
5
6     public float L1=3f,a=7f,b=10f,L2=2f;
7     public Vector3 0;
8     public Color Base, arms, forearms, platform;
9     public float theta1=60f,theta2=60f,theta3=60f;
10    public bool controlled=false;
11    public bool calculateWorkspace=true;
12

```

```

13 public Vector3 Ocontrolled;
14 public bool valid=true;
15
16 private const float rad = 2f*Mathf.PI/360f;
17 private LineRenderer[] Lines;
18 private int line;
19 //plataforma base
20 private Vector3 P,D1,D2,D3;
21 //braço
22 private Vector3 J1,J2,J3;
23 //plataforma pinça
24 private Vector3 E1,E2,E3;
25
26 void Start () {
27     line=0;
28     Lines = GetComponentsInChildren <LineRenderer> ();
29     for (int i=0; i<Lines.Length; ++i)Lines[i].enabled = false;
30 }
31
32 void addLine (Vector3 a, Vector3 b, Color col){
33     Lines[line].SetVertexCount(2);
34     Lines[line].enabled=true;
35     Lines[line].SetColors(col,col);
36     Lines[line].SetPosition(0,a);
37     Lines[line].SetPosition(1,b);
38     line++;
39 }
40 void addTriangle (Vector3 a, Vector3 b, Vector3 c, Color col){
41     Lines[line].SetVertexCount(4);
42     Lines[line].enabled=true;
43     Lines[line].SetColors(col,col);
44     Lines[line].SetPosition(0,a);
45     Lines[line].SetPosition(1,b);
46     Lines[line].SetPosition(2,c);
47     Lines[line].SetPosition(3,a);
48     line++;
49 }
50 void Update () {
51     if (controlled)O=Ocontrolled;
52     line=0;
53     P=transform.position;
54     drawBase ();
55     drawPlatform();
56     setAngles ();
57     drawArms ();
58     drawForearm();
59 }
60 public void setAngles (){
61     if (calculateWorkspace)O=Ocontrolled;
62     float cos60=Mathf.Cos (60*rad);
63     float sin60=Mathf.Sin (60*rad);
64
65     float x0,y0,z0;
66     x0=0.z-P.z;
67     y0=0.y-P.y;
68     z0=0.x-P.x;
69
70     float x1,y1,z1;
71     x1=x0+L2-L1;
72     y1=y0;
73     z1=z0;
74     theta1=singleAngle (x1,y1,z1,a,b);
75
76     float x2,y2,z2;
77     x2=z0*sin60-x0*cos60+L2-L1;
78     y2=y0;
79     z2=-z0*cos60-x0*sin60;
80     theta2=singleAngle (x2,y2,z2,a,b);
81
82     float x3,y3,z3;
83     x3=-z0*sin60-x0*cos60+L2-L1;
84     y3=y0;
85     z3=-z0*cos60+x0*sin60;
86     theta3=singleAngle (x3,y3,z3,a,b);
87
88

```

```

89     valid=(!float.IsNaN(theta1) && !float.IsNaN(theta2) && !float.IsNaN(theta3));
90 }
91 float singleAngle (float x0, float y0, float z0, float r1, float r2){
92     float n = r2 * r2 - r1 * r1 - z0 * z0 - x0 * x0 - y0 * y0;
93     float y = -Mathf.Sqrt ((n*n+4*r1*x0*x0)/(4*(x0*x0-y0*y0)));
94     int signe=1;
95     if ((r2*r2-(y0+r1)*(y0+r1))<(x0*x0+z0*z0) && x0<0)signe = signe * -1;
96     float x = Mathf.Sqrt(r1 * r1 - y * y)*signe;
97     return -Mathf.Atan2 (y,x)*180/Mathf.PI;
98 }
99 void drawBase(){
100     D1.Set(0,0,L1);
101     D1+=P;
102     D2.Set(L1*Mathf.Sin(60f*rad),0,-L1*Mathf.Cos(60f*rad));
103     D2+=P;
104     D3.Set(-L1*Mathf.Sin(60f*rad),0,-L1*Mathf.Cos(60f*rad));
105     D3+=P;
106     addTriangle (D1,D2,D3, Base);
107 }
108 void drawArms(){
109     J1.Set (0,-a*Mathf.Sin (theta1*rad),a*Mathf.Cos(theta1*rad));
110     J1+=D1;
111     addLine(D1,J1, arms);
112     J2.Set (a*Mathf.Cos(theta2*rad)*Mathf.Sin(60f*rad),-a*Mathf.Sin (theta2*rad),-a*Mathf.Cos
        (60f*rad)*Mathf.Cos(theta2*rad));
113     J2+=D2;
114     addLine(D2,J2, arms);
115     J3.Set (-a*Mathf.Cos(theta3*rad)*Mathf.Sin(60f*rad),-a*Mathf.Sin (theta3*rad),-a*Mathf.Cos
        (60f*rad)*Mathf.Cos(theta3*rad));
116     J3+=D3;
117     addLine(D3,J3, arms);
118 }
119 void drawPlatform(){
120     E1.Set(0,0,L2);
121     E1+=0;
122     E2.Set(L2*Mathf.Sin(60f*rad),0,-L2*Mathf.Cos(60f*rad));
123     E2+=0;
124     E3.Set(-L2*Mathf.Sin(60f*rad),0,-L2*Mathf.Cos(60f*rad));
125     E3+=0;
126     addTriangle (E1,E2,E3, platform);
127 }
128 void drawForearm(){
129     addLine (J1,E1, forearms);
130     addLine (J2,E2, forearms);
131     addLine (J3,E3, forearms);
132 }
133 }

```

- **Movements:** Modifica els paràmetres públics del robot per provar moviments determinats.

```

1  using UnityEngine;
2  using System.Collections;
3  using System.IO;
4  using System;
5
6  public class Movements : MonoBehaviour {
7
8      public float height=10f;
9      public float radius=20f;
10     public float w=3f;
11     public RobotCreator robot;
12     private float t=0f;
13     public string path;
14
15     public float rangexz, rangey, steps;
16
17     public void Start(){
18         StartCoroutine(calc());
19     }
20
21     public IEnumerator calc(){
22         if (robot.calculateWorkspace){
23
24             string content="";
25             Transform trans = robot.GetComponentInParent<Transform>();
26

```

```

27     for (float dy=0; dy<rangey; dy+=rangey/steps){
28         for (float dx=-rangexz/2; dx<=rangexz/2; dx+=rangexz/steps){
29             for (float dz=-rangexz/2; dz<=rangexz/2; dz+=rangexz/steps){
30                 Vector3 d= new Vector3 (dx,-dy, dz);
31                 robot.Ocontrolled=trans.position+d;
32                 robot.setAngles();
33                 if (robot.valid){
34                     Vector3 res=robot.Ocontrolled-trans.position;
35                     content = content + res.x.ToString("F2") + "□" +
36                         res.z.ToString("F2") + "□" +
37                         res.y.ToString("F2") + Environment.NewLine;
38                 }
39                 yield return new WaitForSeconds(0.000000f);
40             }
41         }
42     }
43
44     System.IO.File.WriteAllText(path, content);
45     robot.calculateWorkspace=false;
46 }
47 yield return new WaitForSeconds(2);
48 }
49
50 void Update () {
51     t+=Time.deltaTime;
52     if ((w*t)>(2*Mathf.PI))t-=2*Mathf.PI/w;
53     if (!robot.calculateWorkspace)robot.Ocontrolled.Set (radius*Mathf.Sin (w*t),height,radius*
        Mathf.Cos (w*t));
54 }
55 }

```