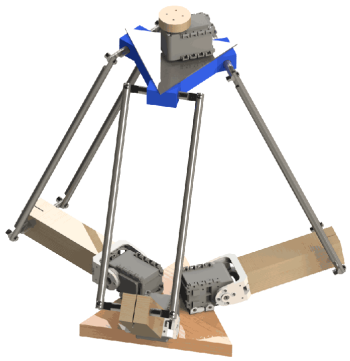
Informe Projecte I

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

23 de juny de 2015



Índex

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.

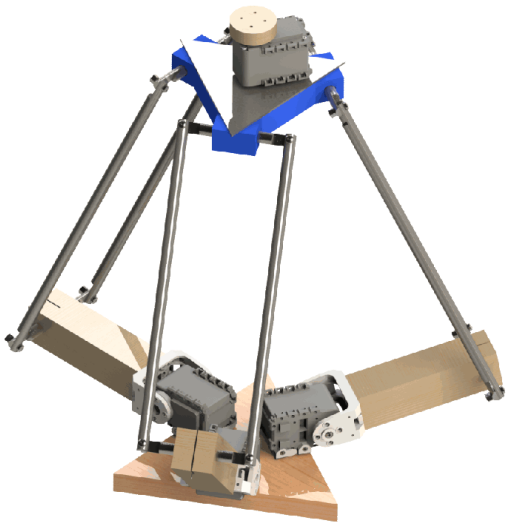


Figura 1: Assemblatge general

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.

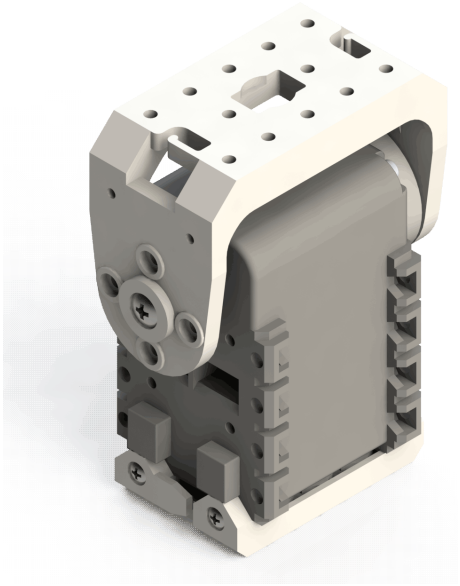


Figura 2: Imatge 3D del servomotor

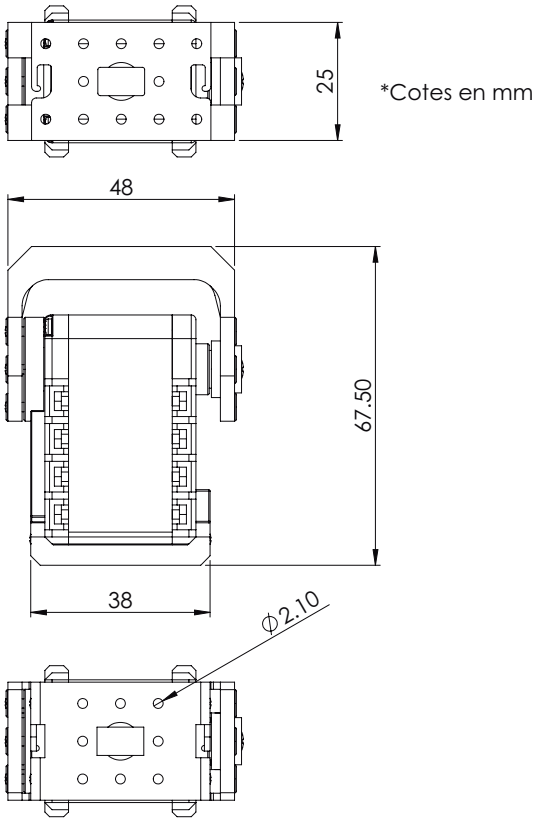


Figura 3: Plànol servo amb accessoris

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.

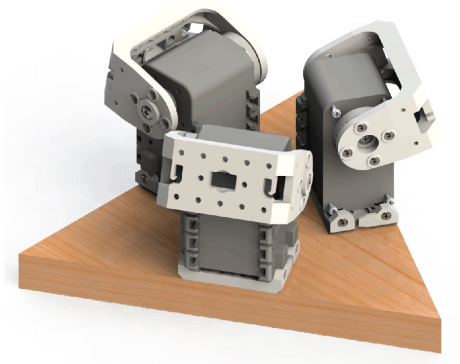


Figura 4: Plataforma amb els servos posats

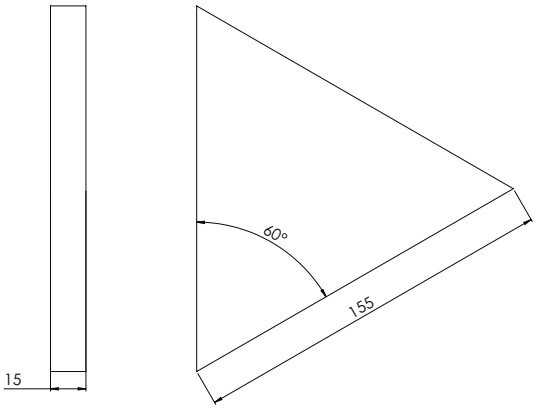


Figura 5: Plànol plataforma servos

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

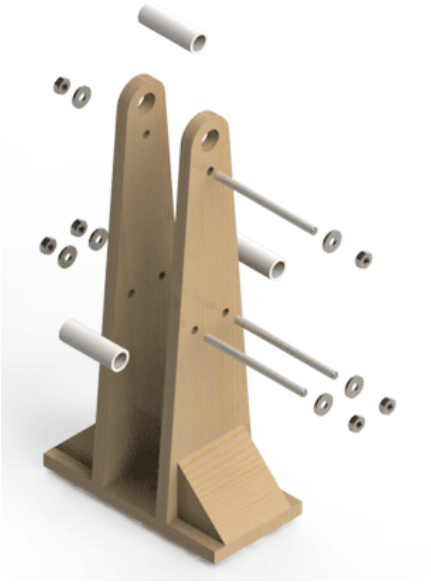


Figura 7: Muntatge del braç

2.3.1 Estructura

L'estructura està feta amb llistons de fusta. S'ha utilitzat un llistó de 45º 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.

Figura 8: Estructura del braç

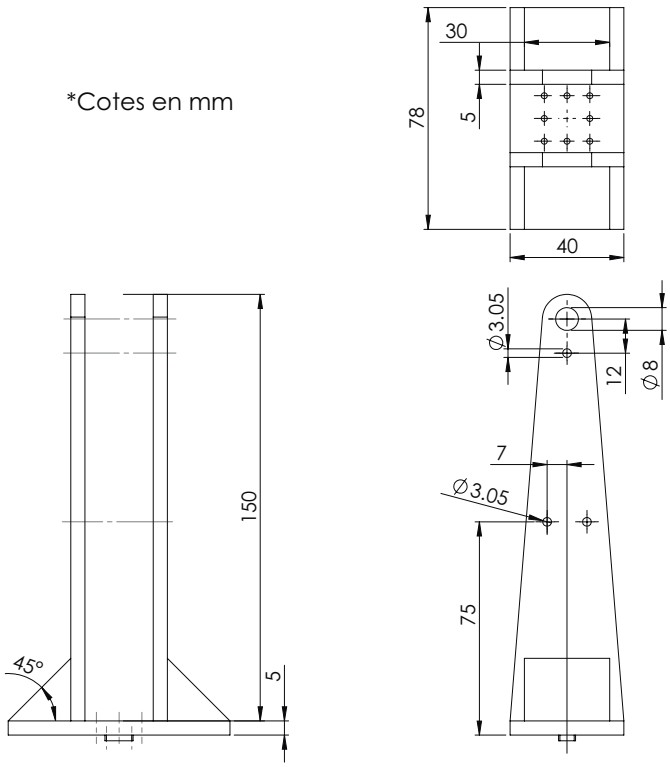


Figura 9: Plànol de les mides del braç

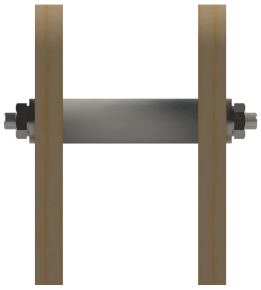


Figura 10: Estabilitzador muntat

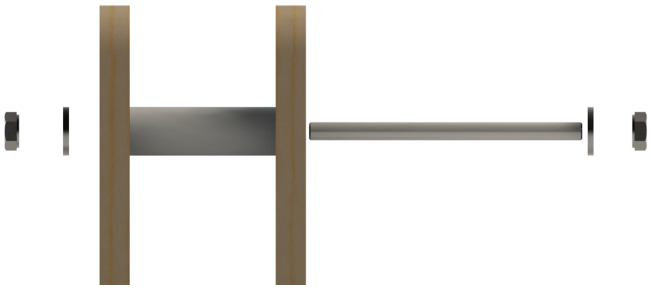


Figura 11: Muntatge estabilitzador

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

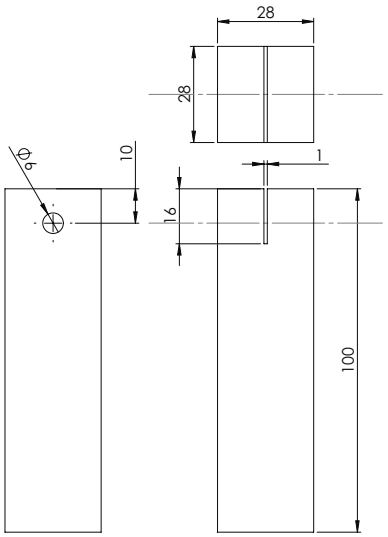


Figura 13: Nou braç

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.

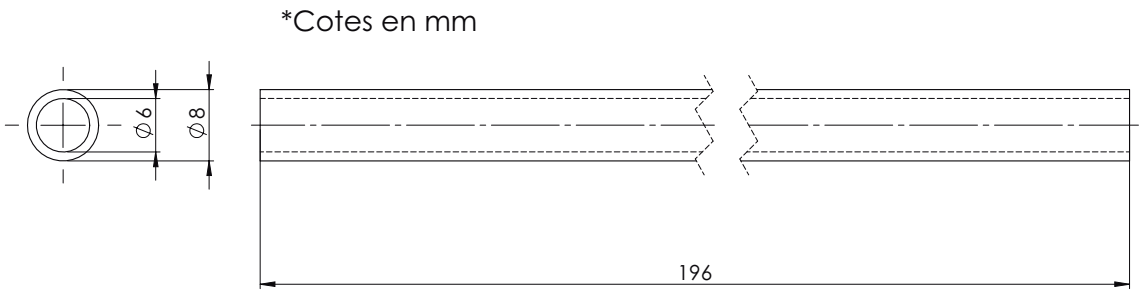


Figura 16: Plànol de la vara

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.

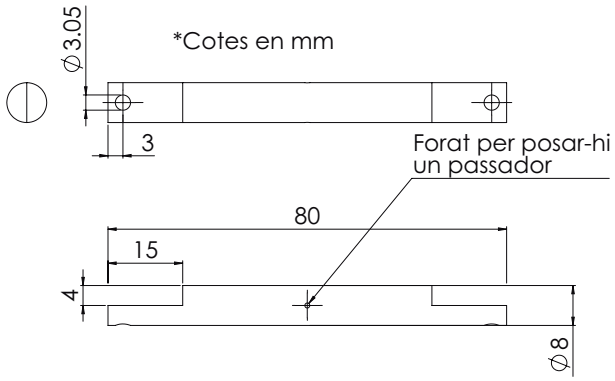


Figura 17: Plànol de l'eix

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.

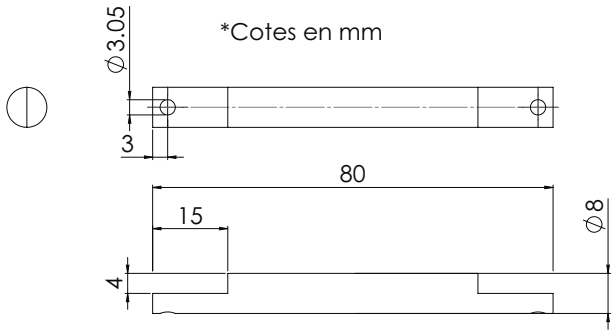
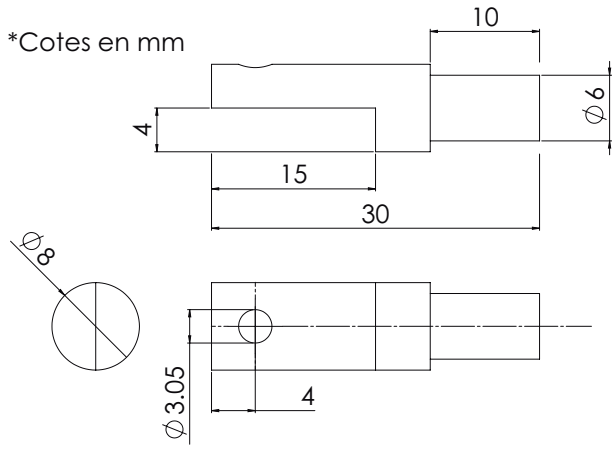


Figura 18: Plànol de l'eix de la pinça

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.

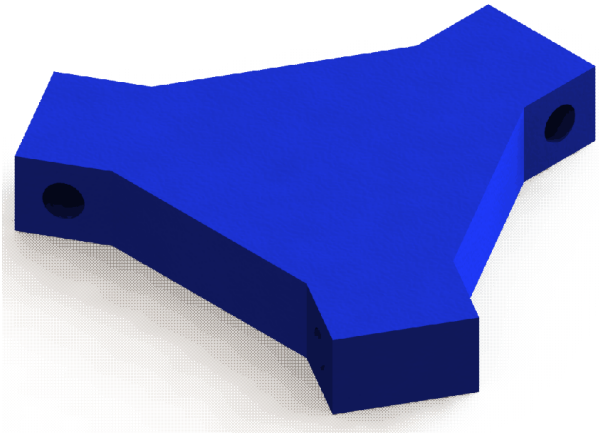


Figura 19: Representació 3D del suport de la pinça

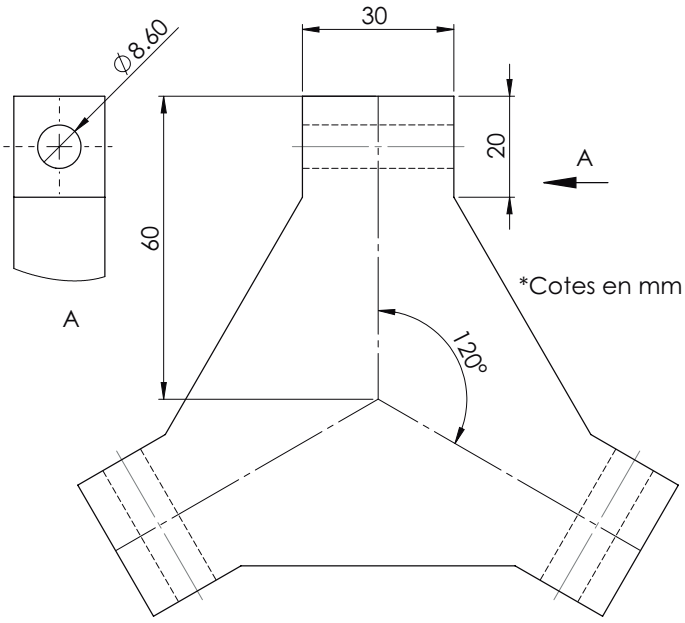


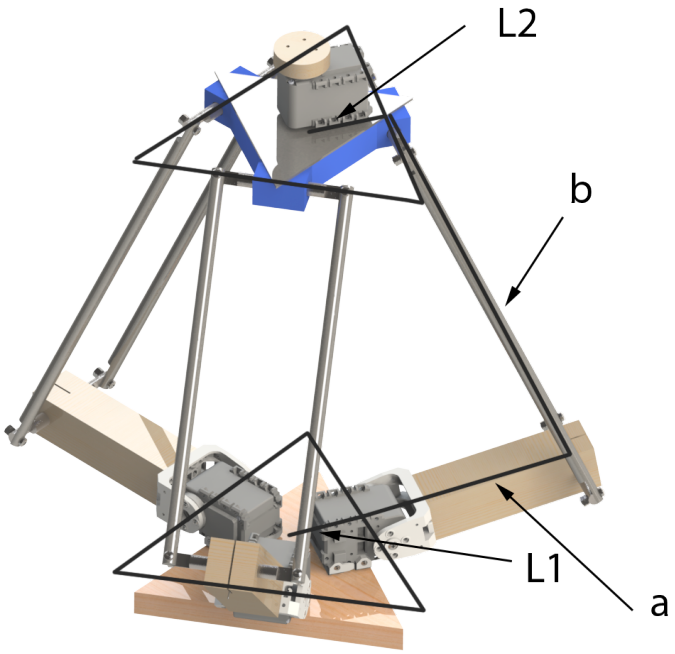
Figura 20: Plànol del suport de la pinça

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.

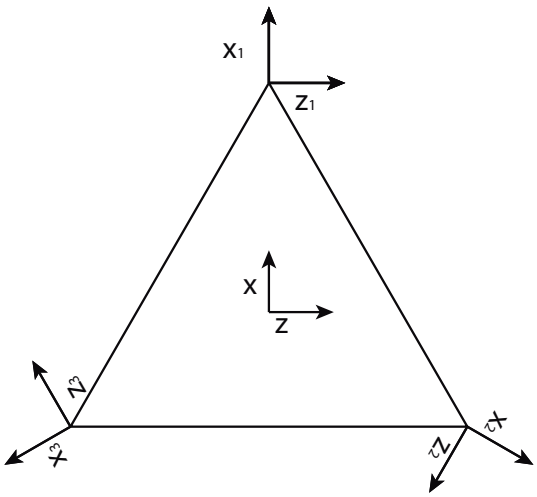
L1 és la distància entre el centre de la base als eixos dels motors.

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.

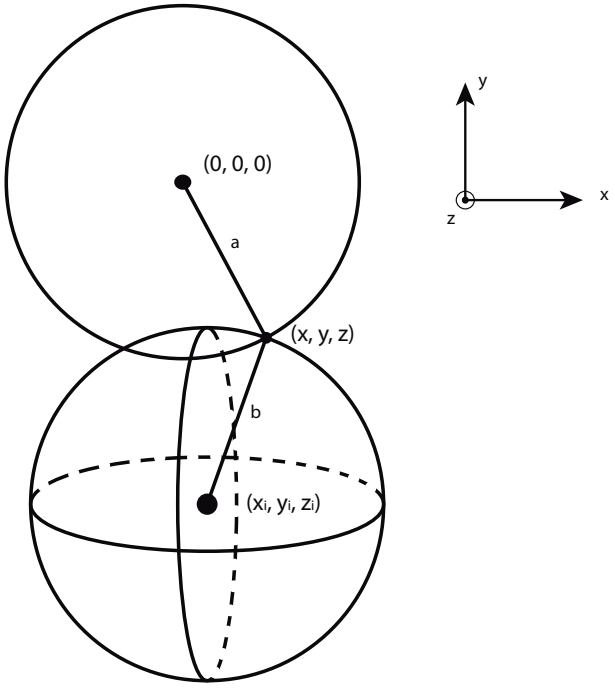
$$\{x_1, y_1, z_1\} = \{x_0 + L2 - L1, y_0, z_0\}$$
$$\{x_2, y_2, z_2\} = \{z_0 sin(60) - x_0 cos(60) + L2 - L1, y_0, -z_0 cos(60) - x_0 sin(60)\}$$
$$\{x_3, y_3, z_3\} = \{-z_0 sin(60) - x_0 cos(60) + L2 - L1, y_0, -z_0 cos(60) + x_0 sin(60)\}$$



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.

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

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

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

$$4a^2x_i^2 - 4y^2x_i^2 = n^2 + 4yy_in + 4y^2y_i^2$$

$$y^2(x_i^2 + y_i^2) + y(y_in) + (\frac{n^2}{4} - a^2x_i^2) = 0$$

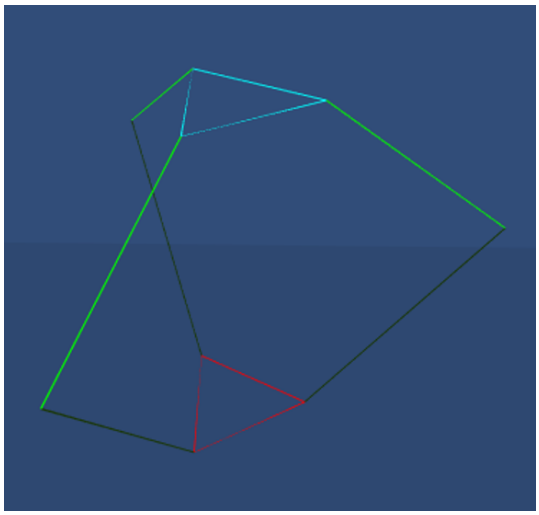
$$y = \frac{-y_i \pm \sqrt{y_i^2n^2 - 4(x_i^2 + y_i^2)(\frac{n^2}{4} - a^2x_i^2)}}{2(x_i^2 + y_i^2)}$$

$$x = \pm\sqrt{a^2 - y^2}$$

$$\text{Finalment, } \theta = atan(\frac{y}{x})$$

3.2 Comprovació dels càlculs

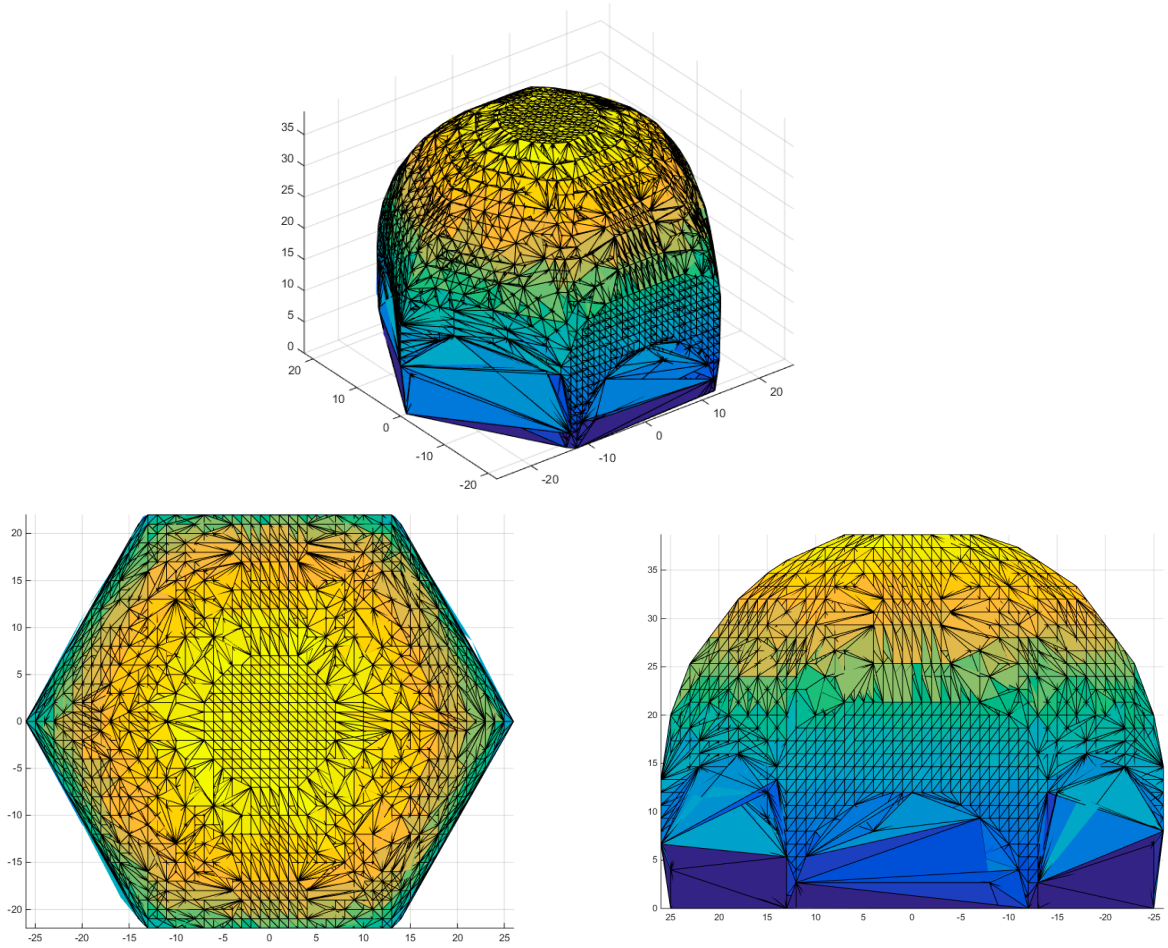
Per poder provar les nostres funcions sense trencar el robot, es va decidir fer una simulació en del Robot en Unity. La simulació és molt simple, Unity s'encarrega de dibuixar un esquema 3D del robot utilitzant línies a partir de la posició desitjada de la plataforma i els angles del robot calculats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

```
% Rep com a parmetres:
% - x, y, z de la posicio a calcular en la base adequada
% - a la llargada del brac
% - b la llargada de l'avantbrac
function angle = singleAngle(x0, y0, z0, a, b)

n = b^2 - a^2 - z0^2 - x0^2 - y0^2;
root = sqrt (n^2*y0^2 - 4*(x0^2 + y0^2)*(-x0^2*a^2 + n^2/4));

if (x0 < 0)
    root = root * -1;
end

y = (-n*y0 + root ) / (2*(x0^2 + y0^2));

sign = 1;
if ((b^2 - (y0 + a)*(y0 + a)) < (x0^2 + z0^2) && x0 < 0)
    sign = -1;
end

x = sqrt(a^2 - y^2)*sign;

if (isreal(x) && isreal(y))
    angle = atan2(y, x);
else
    angle = NaN;
end

end
```

3.4.2 Càlcul de tots els angles

```
% Rep com a parametre la posicio x, y, z on es vol saber els tres angles
% del robot que es el que retorna la funcio.
function D = setAngles(x, y, z)

% Mides del robot
%a = 17.233;     % Llargada brac
a = 12;
b = 22.648;     % Llargada avantbrac
L1 = 6.374;     % Distancia al centre del triangle de la base
L2 = 6;         % Distancia al centre del triangle de la pinca

% Definicions de constants
sin60 = sqrt(3)/2;
cos60 = 1/2;

% Calcul primer angle
x1 = x + L2 - L1;
y1 = -z;
z1 = y;
D(1) = singleAngle(x1,y1,z1, a, b);

% Calcul segon angle
x2 = y*sin60 - x*cos60 + L2 - L1;
y2 = -z;
z2 = -y*cos60 - x*sin60;
D(2) = singleAngle(x2,y2,z2, a, b);

% Calcul tercer angle
x3 = -y*sin60 - x*cos60 + L2 - L1;
y3 = -z;
z3 = -y*cos60 + x*sin60;
D(3) = singleAngle(x3,y3,z3, a, b);

for  i = 1:3
    D(i) = 150 - D(i)*180/pi;
end

end
```

3.4.3 Càlcul del rang de treball

```
% Es van provant tots els punts a l'espai definit per [-x, x], [-y, y] i
% [0, z] i si no retorna NaN (Not a Number) es considera com a valid i es
% guarda als vectors, tot seguit es genera una superficie en 3D junt amb el
% grafic.
function M = calcWorkspace(x, y, z, steps)
i = 1;
for dz = 0:z/(2*steps):z
   for dx = -x:x/steps:x
      for dy = -y:y/steps:y
         D = setAngles(dx, dy, dz);
         if (~isnan(D))
            bool = 1;
            for j=1:3
               if(D(j) > 250 || D(j) < 80)
                   bool = 0;
               end
            end

            if(bool == 1)
                X(i) = dx;
                Y(i) = dy;
                Z(i) = dz;
                i = i + 1;
            end
         end
      end
   end
end

tri = delaunay(X, Y, Z);
trisurf(tri,X, Y, Z, 'EdgeColor', 'black', 'LineWidth', 0.05);
axis('equal')

M(:, 1) = X;
M(:, 2) = Y;
M(:, 3) = Z;
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 robotï 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

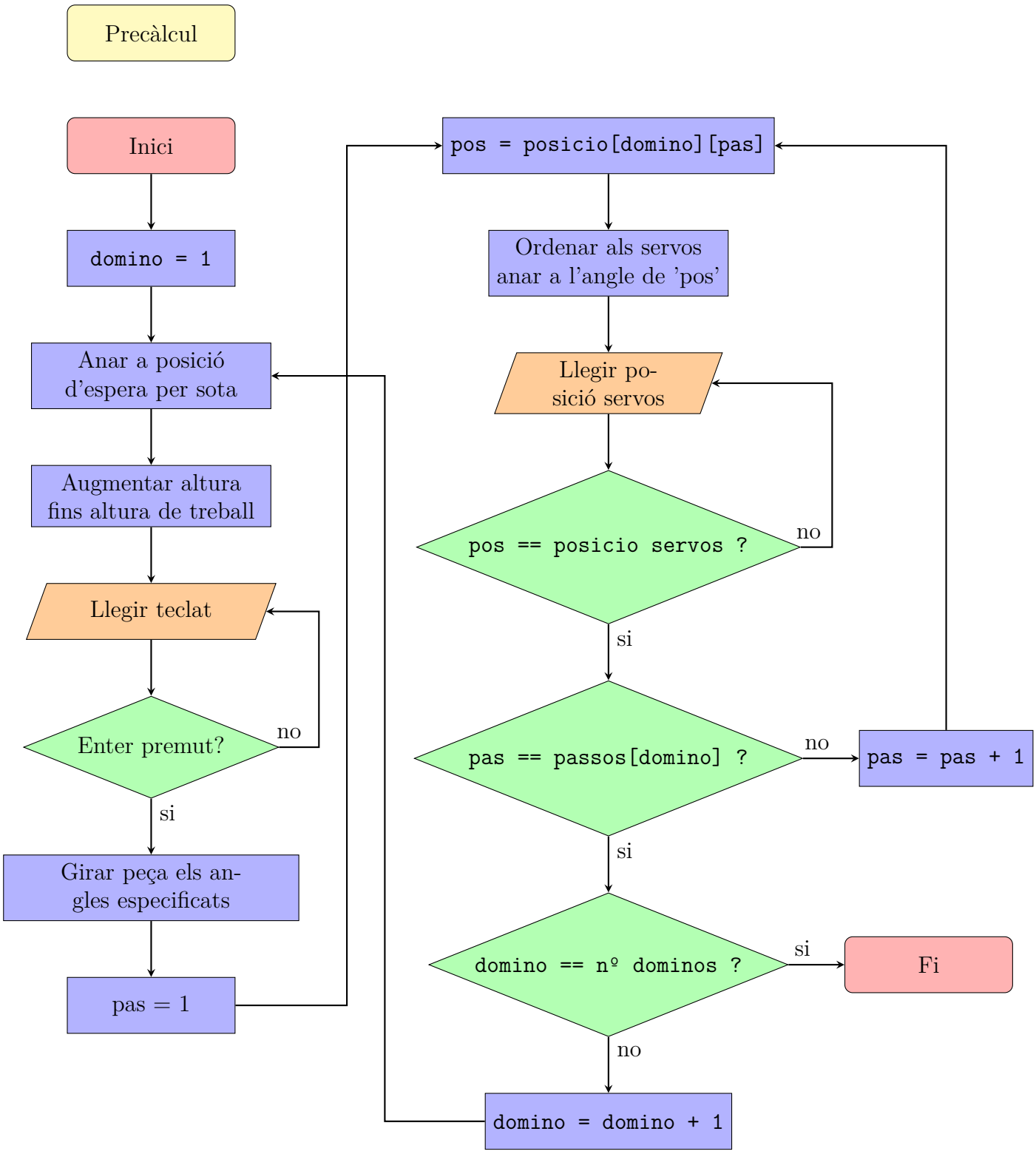
Per fer el control automàtic s'ha utilitzat Qt 5.4 i també la API en C de Dynamixel de comunicació amb els servos. En aquest informe no es detalla molt aquesta part ja que a l'annex hi ha la documentació completa.
Al programa se li carrega un arxiu que conté totes les peces de dominó (extensió `.df`) 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 Us 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`.



4.2 Unity

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    //Debug.Log (centered);
42    if (!onlyZoom) update_objective ();
43    update_angles ();
44    if (!onlyZoom) update_transform();
45  }
46
47
48  void update_objective (){
49    if (Input.GetKeyDown (KeyCode.Mouse2)){
50      Ray CamRay = Camera.main.ScreenPointToRay(Input.mousePosition);
51      Debug.DrawRay (CamRay.origin, CamRay.direction);
52      RaycastHit floorHit;
53      if (Physics.Raycast (CamRay, out floorHit, camRayLength, 1)) {
54        switchingobjective=true;
55        centered=true;
56        center=floorHit.collider.gameObject;
57      }
58    }
59    if (Input.GetKey(KeyCode.A)){
60      if (centered){
61        centered=false;
62        switchingobjective=false;
63        ObjectiveDesired=obj_pos;
64      }
65      Vector3 desp=new Vector3 (Mathf.Sin(theta*rad),0,-Mathf.Cos(theta*rad));
66      obj_pos+=desp*movementSpeed*Time.fixedDeltaTime;
67    }
68    if (Input.GetKey(KeyCode.D)){
69      if (centered){
70        centered=false;
71        switchingobjective=false;
72        ObjectiveDesired=obj_pos;
73      }
74      Vector3 desp=new Vector3 (-Mathf.Sin(theta*rad),0,Mathf.Cos(theta*rad));
75      obj_pos+=desp*movementSpeed*Time.fixedDeltaTime;
76    }
77    if (Input.GetKey(KeyCode.W)){
78      if (centered){
79        centered=false;
80        switchingobjective=false;
81        ObjectiveDesired=obj_pos;
```

```
82        }
83        Vector3 desp=new Vector3 (-Mathf.Cos(theta*rad),0,-Mathf.Sin(theta*rad));
84        obj_pos+=desp*movementSpeed*Time.fixedDeltaTime;
85      }
86      if (Input.GetKey(KeyCode.S)){
87        if (centered){
88          centered=false;
89          switchingobjective=false;
90          ObjectiveDesired=obj_pos;
91        }
92        Vector3 desp=new Vector3 (Mathf.Cos(theta*rad),0,Mathf.Sin(theta*rad));
93        obj_pos+=desp*movementSpeed*Time.fixedDeltaTime;
94      }
95      if (centered)ObjectiveDesired=center.transform.position;
96      if (switchingobjective){
97        Vector3 error=ObjectiveDesired-obj_pos;
98        obj_pos=Vector3.Lerp(obj_pos, ObjectiveDesired, objectiveSwitchSpeed*Time.fixedDeltaTime
              /(Mathf.Pow (error.magnitude,0.5f)));
99
100       //Debug.Log(error.magnitude);
101       if (error.magnitude<0.02)switchingobjective=false;
102     }
103     else{
104       if (centered)obj_pos=ObjectiveDesired;
105       Debug.DrawLine (transform.position, obj_pos);
106     }
107   }
108   void update_angles (){
109     //Debug.Log (desiredDistance);
110     //distance = Mathf.Clamp(distance - Input.GetAxis("Mouse ScrollWheel")*zoomspeed*Time.
          fixedDeltaTime, distanceMin, distanceMax);
111     desiredDistance = Mathf.Clamp(desiredDistance - Input.GetAxis("Mouse␣ScrollWheel")*
          zoomspeed*Time.fixedDeltaTime, distanceMin, distanceMax);
112     distance=Mathf.Lerp (distance,desiredDistance,10*Time.fixedDeltaTime);
113     size -= zoomspeed*Time.fixedDeltaTime*Input.GetAxis("Mouse␣ScrollWheel");
114     size = Mathf.Clamp(size,1f,100f);
115     Camera.main.orthographicSize=size;
116
117     if (!onlyZoom) {
118             if (Input.GetKeyDown (KeyCode.Mouse1))
119                 LastMouse = Input.mousePosition;
120             if (Input.GetKey (KeyCode.Mouse1)) {
121                 Vector2 dist = Input.mousePosition;
122                 dist -= LastMouse;
123                 theta -= dist.x * dragSpeed;
124                 phi += dist.y * dragSpeed;
125                 LastMouse = Input.mousePosition;
126             }
127             theta = theta % 360;
128             phi = Mathf.Clamp (phi, 0.05f, 179.95f);
129         }
130   }
131   void update_transform (){
132     Vector3 despl = new Vector3(Mathf.Sin(phi*rad)*Mathf.Cos(theta*rad),
133                                 Mathf.Cos(phi*rad),
134                                 Mathf.Sin(phi*rad)*Mathf.Sin(theta*rad));
135     despl*=distance;
136     transform.position = obj_pos + despl;
137     transform.rotation = Quaternion.LookRotation (-despl);
138   }
139
140 }
```

- `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 forceApplyer;
7    public bool adding = false;
8    public float speed = 10f;
9
10
11   private bool shot = false;
```

```
12
13    // Use this for initialization
14    void Start () {
15    }
16
17    // Update is called once per frame
18    void Update () {
19      if (adding) {
20            Ray camRay = Camera.main.ScreenPointToRay (Input.mousePosition);
21            RaycastHit floorHit;
22
23            if (Physics.Raycast (camRay, out floorHit, 100f, LayerMask.GetMask ("Floor"))) {
24              transform.position = floorHit.point + height*Vector3.up;
25            }
26      }
27
28      if (Input.GetKeyDown (KeyCode.Mouse0))
29              adding = false;
30
31      if (forceApplyer && !shot && Time.timeScale==Editor.timescale) {
32        GetComponent<Rigidbody>().velocity = transform.forward * speed;
33      }
34    }
35
36    void OnCollisionEnter (Collision collision){
37      if (forceApplyer && collision.gameObject.tag == "Player") {
38              gameObject.SetActive(false);
39              //Debug.Log ("lel");
40          }
41    }
42  }
```

- `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    // Use this for initialization
15    void Start () {
16      canvas.SetActive (false);
17    }
18
19    // Update is called once per frame
20    void Update () {
21      canvas.SetActive (false);
22      if (Input.GetKeyDown (KeyCode.Mouse0)) // Començar o parar un dibuix
23      {
24        drawing = !drawing;
25        first = true;
26        lastPosition = new Vector3(100,100,100);
27      }
28
29      if (Input.GetKeyDown (KeyCode.Escape) || Input.GetKeyDown (KeyCode.Return)) // Tornar al
            modo editor
30      {
31        CameraController.onlyZoom = false;
32        Camera.main.orthographic = false;
33        GetComponent<Editor> ().enabled = true;
34
35        canvas.SetActive (true);
36
37        GetComponent<Draw> ().enabled = false;
38      }
39
40      if (Input.GetKeyDown (KeyCode.Delete)) // Eliminar tots els dominos
41      {
```

```
42        GameObject[] pieces = GameObject.FindGameObjectsWithTag ("Player");
43        for (int i = 0; i<pieces.Length; ++i) Destroy (pieces[i]);
44      }
45
46      if (drawing) // Instantiate de dominos si estem dibuixant
47      {
48        Ray camRay = Camera.main.ScreenPointToRay (Input.mousePosition);
49        RaycastHit floorHit;
50
51        if (Physics.Raycast (camRay, out floorHit, 1000f, LayerMask.GetMask ("Floor")))
52        {
53          if (Vector3.Distance(floorHit.point, lastPosition)>= distance)
54          {
55            if (!first) {
56              lastDomino.transform.rotation = Quaternion.LookRotation(floorHit.point-
                    lastPosition);
57              lastDomino = Instantiate (dominoPiece, lastPosition+(floorHit.point-lastPosition).
                    normalized*distance + 1.2f*dominoPiece.transform.localScale.y*Vector3.up,
                    Quaternion.LookRotation(floorHit.point - lastPosition)) as GameObject;
58              lastPosition = lastPosition+(floorHit.point-lastPosition).normalized*distance;
59              Debug.Log (floorHit.point.y);
60            }
61            else {
62              first = false;
63              lastDomino = Instantiate (dominoPiece, floorHit.point + 1.2f*dominoPiece.transform
                    .localScale.y*Vector3.up, Quaternion.LookRotation(Vector3.forward)) as
                    GameObject;
64              Debug.Log (floorHit.point);
65              lastPosition=floorHit.point;
66            }
67          }
68        }
69      }
70    }
71  }
```

- `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    private Vector2 LastMouse;
30    private Vector3 toLastMouseR;
31    private string lastMode;
32    private Quaternion RXinitial, RYinitial, RZinitial;
33
34    // Use this for initialization
35    void Start () {
36      Physics.gravity = 19.6f * Vector3.down;
37      Time.timeScale=0f;
38      selected = Camera.main.GetComponent<CameraController>().center;
```

```
39      selectedExists = false;
40      camRayLength = GameObject.FindGameObjectWithTag("MainCamera").GetComponent<
            CameraController>().camRayLength;
41      //lastColor = selected.GetComponentInChildren<Renderer>().material.color;
42      //selected.GetComponentInChildren<Renderer>().material.color = selectedColor;
43      if (relativeRotation){
44        RXinitial=RX.transform.rotation;
45        RYinitial=RY.transform.rotation;
46        RZinitial=RZ.transform.rotation;
47      }
48      X.SetActive (true); Y.SetActive (true); Z.SetActive (true); RX.SetActive (true); RY.
            SetActive (true); RZ.SetActive (true);
49      X.GetComponentInChildren<Renderer>().material.color = Color.red;
50      Y.GetComponentInChildren<Renderer>().material.color = Color.green;
51      Z.GetComponentInChildren<Renderer>().material.color = Color.blue;
52      RX.GetComponentInChildren<Renderer>().material.color = Color.red;
53      RY.GetComponentInChildren<Renderer>().material.color = Color.green;
54      RZ.GetComponentInChildren<Renderer>().material.color = Color.blue;
55    }
56
57    // Update is called once per frame
58    void Update () {
59
60      //bola.transform.position=pos;
61      //Debug.DrawRay (RX.transform.position, Vector3.Cross(RX.transform.forward,Vector3.up).
            normalized*4);
62
63
64      align_tools ();
65      update_mode ();
66      update_selected();
67      align_tools ();
68    }
69
70    void align_tools (){
71      X.SetActive(mode=="t" & selectedExists);
72      Y.SetActive(mode=="t" & selectedExists);
73      Z.SetActive(mode=="t" & selectedExists);
74      RX.SetActive(mode=="r" & selectedExists);
75      RY.SetActive(mode=="r" & selectedExists);
76      RZ.SetActive(mode=="r" & selectedExists);
77      if (selectedExists) {
78        X.transform.position = selected.transform.position;
79        Y.transform.position = selected.transform.position;
80        Z.transform.position = selected.transform.position;
81        RX.transform.position = selected.transform.position;
82        RY.transform.position = selected.transform.position;
83        RZ.transform.position = selected.transform.position;
84        if (relativeRotation){
85          RX.transform.rotation = selected.transform.rotation*RXinitial;
86          RY.transform.rotation = selected.transform.rotation*RYinitial;
87          RZ.transform.rotation = selected.transform.rotation*RZinitial;
88        }
89      }
90    }
91
92    void update_mode (){
93      if (Time.timeScale==0){
94        if (Input.GetKeyDown(KeyCode.T)){
95          if (mode=="t")mode="-";
96          else if (selectedExists) mode="t";
97        }
98        if (Input.GetKeyDown(KeyCode.R)){
99          if (mode=="r")mode="-";
100         else if (selectedExists) mode="r";
101       }
102     }
103   }
104
105   void update_selected (){
106     if (Input.GetKeyDown (KeyCode.Delete)) {
107       Destroy (selected);
108       selectedExists = false;
109       mode = "-";
110       Camera.main.GetComponent <CameraController>().centered = false;
111     }
```

```
112     //Debug.Log(dragging);
113     if (Input.GetKeyDown (KeyCode.Mouse0)){
114       Ray CamRay = Camera.main.ScreenPointToRay(Input.mousePosition);
115       //Debug.DrawRay (CamRay.origin, CamRay.direction);
116       RaycastHit floorHit;
117       if (Physics.Raycast (CamRay, out floorHit, camRayLength, 1)) {
118         if (floorHit.collider.gameObject.CompareTag("Axis")){
119           LastMouse = Input.mousePosition;
120           dragging = true;
121           char auxc = floorHit.collider.gameObject.name[0];
122           if (auxc == 'X')draggingAxis = X;
123           if (auxc == 'Y')draggingAxis = Y;
124           if (auxc == 'Z')draggingAxis = Z;
125         }
126         else if (floorHit.collider.gameObject.CompareTag("Rotator")){
127           dragging = true;
128           char auxc = floorHit.collider.gameObject.name[0];
129           if (auxc == 'X')draggingRotator = RX;
130           if (auxc == 'Y')draggingRotator = RY;
131           if (auxc == 'Z')draggingRotator = RZ;
132           Ray MouseRay = Camera.main.ScreenPointToRay(Input.mousePosition);
133           float t=Vector3.Dot (draggingRotator.transform.position,draggingRotator.transform.
                  forward)-Vector3.Dot (draggingRotator.transform.forward,MouseRay.origin);
134           t=t/Vector3.Dot (draggingRotator.transform.forward,MouseRay.direction);
135           toLastMouseR=MouseRay.origin+t*MouseRay.direction-draggingRotator.transform.position
                  ;
136         }
137         else{
138           if (selectedExists) selected.GetComponentInChildren<Renderer>().material.color =
                  lastColor;
139           if (floorHit.collider.gameObject.CompareTag("EditorOnly")) selected = floorHit.
                  collider.transform.parent.gameObject;
140           else selected = floorHit.collider.gameObject;
141           //Debug.Log(selected.name);
142           lastColor = selected.GetComponentInChildren<Renderer>().material.color;
143           Renderer[] letsColor = selected.GetComponentsInChildren<Renderer>();
144           for (int i=0; i<letsColor.Length; ++i) letsColor[i].material.color = selectedColor;
145           selectedExists = true;
146         }
147       }
148     }
149     if (Input.GetKeyUp(KeyCode.Mouse0))dragging = false;
150
151     if (Input.GetKey (KeyCode.Mouse0) && dragging && mode=="t"){
152       Vector2 despMouse = Input.mousePosition;
153       despMouse -= LastMouse;
154       LastMouse=Input.mousePosition;
155       Vector2 axisScreenTip = Camera.main.WorldToScreenPoint(draggingAxis.transform.position+
              draggingAxis.transform.up*arrowlength);
156       Vector2 axisScreenBase = Camera.main.WorldToScreenPoint(draggingAxis.transform.position)
              ;
157       Vector2 axisScreen = axisScreenTip-axisScreenBase;
158       /*Debug.Log (' ');
159       Debug.Log (axisScreenTip);
160       Debug.Log (axisScreenBase);
161       Debug.Log ('=');
162       Debug.Log (axisScreen);*/
163       float res = Vector2.Dot (despMouse,axisScreen)/Mathf.Pow(axisScreen.magnitude,2);
164       selected.transform.position+=draggingAxis.transform.up*res*translateSpeed;
165     }
166     if (Input.GetKey (KeyCode.Mouse0) && dragging && mode=="r"){
167       Ray MouseRay = Camera.main.ScreenPointToRay(Input.mousePosition);
168       float t=Vector3.Dot (draggingRotator.transform.position,draggingRotator.transform.
              forward)-Vector3.Dot (draggingRotator.transform.forward,MouseRay.origin);
169       t=t/Vector3.Dot (draggingRotator.transform.forward,MouseRay.direction);
170       Vector3 toPos=MouseRay.origin+t*MouseRay.direction-draggingRotator.transform.position;
171       float dAngle = Vector3.Angle(toLastMouseR , toPos);
172       if (Vector3.Dot (draggingRotator.transform.forward,Vector3.Cross (toLastMouseR,toPos))
              <0)dAngle=-dAngle;
173       toLastMouseR=toPos;
174       selected.transform.Rotate(draggingRotator.transform.forward*dAngle,Space.World);
175     }
176   }
177
178   // CANVAS FUNCTIONS
179   public void PausePlay (){
```

```
180       //Debug.Log (Time.timeScale);
181       if (Time.timeScale == timescale) {
182               Time.timeScale = 0;
183               GameObject.FindGameObjectWithTag ("Pause").GetComponentInChildren <Text> ().text =
                      "Play";
184               mode = lastMode;
185
186         //Debug.Log ("loadd");
187               loadState();
188           }
189       else {
190               if (selectedExists) selected.GetComponentInChildren<Renderer>().material.color =
                      lastColor;
191               selectedExists = false;
192               Time.timeScale = timescale;
193               GameObject.FindGameObjectWithTag("Pause").GetComponentInChildren <Text>().text = "
                      Pause";
194               lastMode = mode;
195               mode = "-";
196
197         string aux=path;
198         path="./temp.df";
199         saveState();
200         path=aux;
201
202         //Debug.Log ("save");
203       }
204     }
205
206     public void AddD (){
207       if (Time.timeScale == 0) {
208               GameObject piece = (GameObject)Instantiate (DominoPiece, DominoPiece.transform.
                      position, DominoPiece.transform.rotation);
209               piece.GetComponent<dominoPosition> ().adding = true;
210           }
211
212     }
213
214     public void AddF (){
215       if (Time.timeScale == 0) {
216               GameObject piece = (GameObject)Instantiate (Force, Force.transform.position, Force
                      .transform.rotation);
217               piece.GetComponent<dominoPosition> ().adding = true;
218           }
219     }
220
221     public void AddS (){
222       if (Time.timeScale == 0) {
223         GameObject piece = (GameObject)Instantiate (Stair, Stair.transform.position, Stair.
              transform.rotation);
224         piece.GetComponent<dominoPosition> ().adding = true;
225       }
226     }
227
228     public void LetsDraw (){
229       if (selectedExists) selected.GetComponentInChildren<Renderer>().material.color = lastColor
            ;
230       selectedExists = false;
231       mode = "-";
232       GetComponent<Draw> ().enabled = true;
233       CameraController.onlyZoom = true;
234       Camera.main.orthographic = true;
235       Camera.main.transform.position = new Vector3 (0, 100, 0);
236       Camera.main.transform.rotation = Quaternion.LookRotation (Vector3.down);
237       GetComponent<Editor> ().enabled = false;
238     }
239
240     public void TimeScale (){
241       timescale = slider.value;
242       Time.timeScale = slider.value;
243     }
244
245     public void saveState (){
246       GameObject[] pieces = GameObject.FindGameObjectsWithTag ("Player");
247       string content = pieces.Length.ToString() + Environment.NewLine;
248       for (int i = 0; i<pieces.Length; ++i)
```

```
249             content = content + pieces [i].transform.position.x.ToString("F2") +"␣" +
250                 pieces [i].transform.position.z.ToString("F2") + "␣" +
251                 pieces[i].transform.rotation.eulerAngles.y.ToString("F2") + Environment.
                        NewLine;
252     System.IO.File.WriteAllText(path, content);
253   }
254
255   void loadState (){
256     GameObject[] pieces = GameObject.FindGameObjectsWithTag("Player");
257     foreach (GameObject piece in pieces) Destroy (piece,0.0f);
258     selectedExists=false;
259     //Debug.Log ("load");
260     StreamReader sr = new StreamReader("./temp.df");
261     string fileContents = sr.ReadToEnd();
262     sr.Close();
263
264     string[] lines = fileContents.Split(new char[] {'\n'});
265     //Debug.Log (lines[2]);
266     for (int i=1; i<lines.Length-1; i++) {
267       string[] vars = lines[i].Split(new char[] {' '});
268       Vector3 newpos= new Vector3 (float.Parse (vars[0]),0.5f,float.Parse(vars[1]));
269       Instantiate (DominoPiece, newpos+ 1.2f*DominoPiece.transform.localScale.y*Vector3.up,
              Quaternion.Euler(0,float.Parse(vars[2]),0));
270     }
271   }
272 }
```

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 O;
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 int signe = 1;
18   private LineRenderer[] Lines;
19   private int line;
20   //plataforma base
21   private Vector3 P,D1,D2,D3;
22   //braç
23   private Vector3 J1,J2,J3;
24
25
26   //plataforma pinça
27   private Vector3 E1,E2,E3;
28
29   void Start () {
30     line=0;
31     Lines = GetComponentsInChildren <LineRenderer> ();
32     for (int i=0; i<Lines.Length; ++i)Lines[i].enabled = false;
33   }
34
35   void addLine (Vector3 a, Vector3 b, Color col){
36     Lines[line].SetVertexCount(2);
37     Lines[line].enabled=true;
38     Lines[line].SetColors(col,col);
39     Lines[line].SetPosition(0,a);
40     Lines[line].SetPosition(1,b);
41     line++;
42   }
```

```
43    void addTriangle (Vector3 a, Vector3 b, Vector3 c, Color col){
44      Lines[line].SetVertexCount(4);
45      Lines[line].enabled=true;
46      Lines[line].SetColors(col,col);
47      Lines[line].SetPosition(0,a);
48      Lines[line].SetPosition(1,b);
49      Lines[line].SetPosition(2,c);
50      Lines[line].SetPosition(3,a);
51      line++;
52    }
53    void Update () {
54      if (controlled)O=Ocontrolled;
55      line=0;
56      P=transform.position;
57      drawBase ();
58      drawPlatform();
59      setAngles ();
60      drawArms ();
61      drawForearm();
62    }
63    public void setAngles (){
64      if (calculateWorkspace)O=Ocontrolled;
65      float cos60=Mathf.Cos (60*rad);
66      float sin60=Mathf.Sin (60*rad);
67
68      float x0,y0,z0;
69      x0=O.z-P.z;
70      y0=O.y-P.y;
71      z0=O.x-P.x;
72
73      float x1,y1,z1;
74      x1=x0+L2-L1;
75      y1=y0;
76      z1=z0;
77      theta1=singleAngle (x1,y1,z1,a,b);
78
79      float x2,y2,z2;
80      x2=z0*sin60-x0*cos60+L2-L1;
81      y2=y0;
82      z2=-z0*cos60-x0*sin60;
83      theta2=singleAngle (x2,y2,z2,a,b);
84
85      float x3,y3,z3;
86      x3=-z0*sin60-x0*cos60+L2-L1;
87      y3=y0;
88      z3=-z0*cos60+x0*sin60;
89      theta3=singleAngle (x3,y3,z3,a,b);
90
91
92      valid=(!float.IsNaN(theta1) && !float.IsNaN(theta2) && !float.IsNaN(theta3));
93      //if (!calculateWorkspace)Debug.Log (valid);
94    }
95    float singleAngle (float x0, float y0, float z0, float r1, float r2){
96      float n = r2 * r2 - r1 * r1 - z0 * z0 - x0 * x0 - y0 * y0;
97      /*float raiz = Mathf.Sqrt (n * n * y0 * y0 - 4 * (x0 * x0 + y0 * y0) * (-x0 * x0 * r1 * r1
            + n * n / 4));
98      if (x0 < 0)raiz = -raiz;
99      float y = (-n*y0 + raiz ) / (2*(x0*x0+y0*y0));*/
100     float y = -Mathf.Sqrt ((n*n+4*r1*x0*x0)/(4*(x0*x0-y0*y0)));
101     Debug.Log(y);
102     int signe=1;
103     if ((r2*r2-(y0+r1)*(y0+r1))<(x0*x0+z0*z0) && x0<0)signe = signe * -1;
104     float x = Mathf.Sqrt(r1 * r1 - y * y)*signe;
105     return -Mathf.Atan2 (y,x)*180/Mathf.PI;
106   }
107   void drawBase(){
108     D1.Set(0,0,L1);
109     D1+=P;
110     D2.Set(L1*Mathf.Sin(60f*rad),0,-L1*Mathf.Cos(60f*rad));
111     D2+=P;
112     D3.Set(-L1*Mathf.Sin(60f*rad),0,-L1*Mathf.Cos(60f*rad));
113     D3+=P;
114     addTriangle (D1,D2,D3, Base);
115   }
116   void drawArms(){
117     J1.Set (0,-a*Mathf.Sin (theta1*rad),a*Mathf.Cos(theta1*rad));
```

```
118       J1+=D1;
119       addLine(D1,J1, arms);
120       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));
121       J2+=D2;
122       addLine(D2,J2, arms);
123       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));
124       J3+=D3;
125       addLine(D3,J3, arms);
126     }
127     void drawPlatform(){
128       E1.Set(0,0,L2);
129       E1+=0;
130       E2.Set(L2*Mathf.Sin(60f*rad),0,-L2*Mathf.Cos(60f*rad));
131       E2+=0;
132       E3.Set(-L2*Mathf.Sin(60f*rad),0,-L2*Mathf.Cos(60f*rad));
133       E3+=0;
134       addTriangle (E1,E2,E3, platform);
135     }
136     void drawForearm(){
137       addLine (J1,E1, forearms);
138       addLine (J2,E2, forearms);
139       addLine (J3,E3, forearms);
140     }
141   }
```

- `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(wot());
19
20      //robot.calculateWorkspace=false;
21    }
22
23    public IEnumerator wot(){
24      if (robot.calculateWorkspace){
25
26        string content="";
27        Transform trans = robot.GetComponentInParent<Transform>();
28
29        for (float dy=0; dy<rangey; dy+=rangey/steps){
30          for (float dx=-rangexz/2; dx<=rangexz/2; dx+=rangexz/steps){
31            for (float dz=-rangexz/2; dz<=rangexz/2; dz+=rangexz/steps){
32              Vector3 d= new Vector3 (dx,-dy, dz);
33              robot.Ocontrolled=trans.position+d;
34              robot.setAngles();
35              if (robot.valid){
36                Vector3 res=robot.Ocontrolled-trans.position;
37                content = content + res.x.ToString("F2") +"␣" +
38                  res.z.ToString("F2") + "␣" +
39                    res.y.ToString("F2") + Environment.NewLine;
40              }
41              yield return new WaitForSeconds(0.000000f);
42            }
43          }
44        }
45
46        System.IO.File.WriteAllText(path, content);
47        robot.calculateWorkspace=false;
```

```
48      }
49      yield return new WaitForSeconds(2);
50    }
51
52    void Update () {
53      t+=Time.deltaTime;
54      if ((w*t)>(2*Mathf.PI))t-=2*Mathf.PI/w;
55      if (!robot.calculateWorkspace)robot.Ocontrolled.Set (radius*Mathf.Sin (w*t),height,radius*
            Mathf.Cos (w*t));
56      //Debug.Log (robot.valid);
57    }
58  }
```