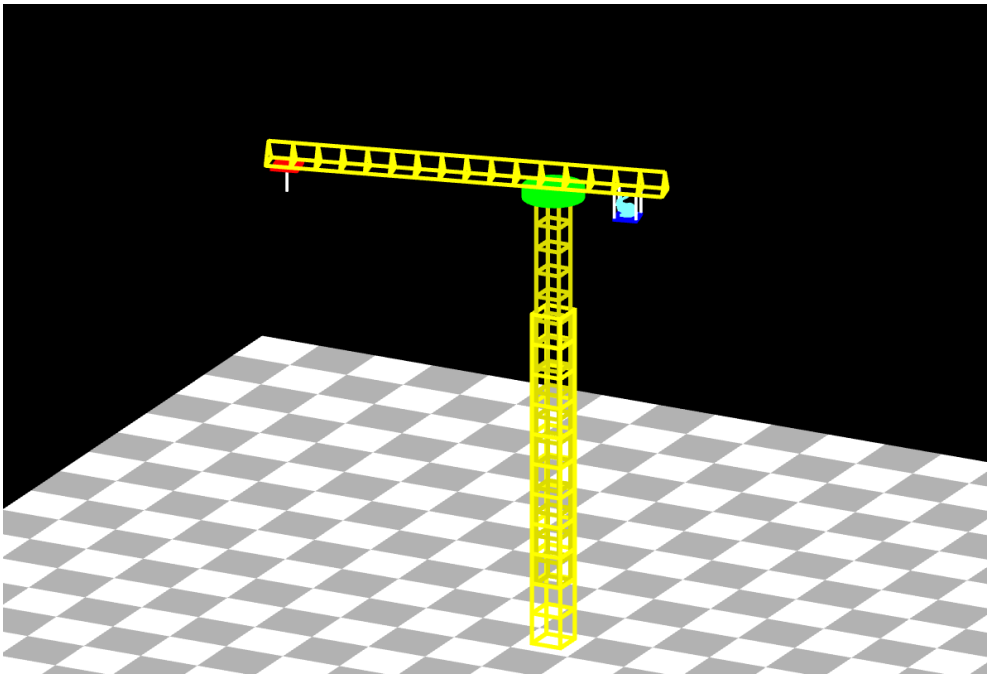
CGI – 23/24

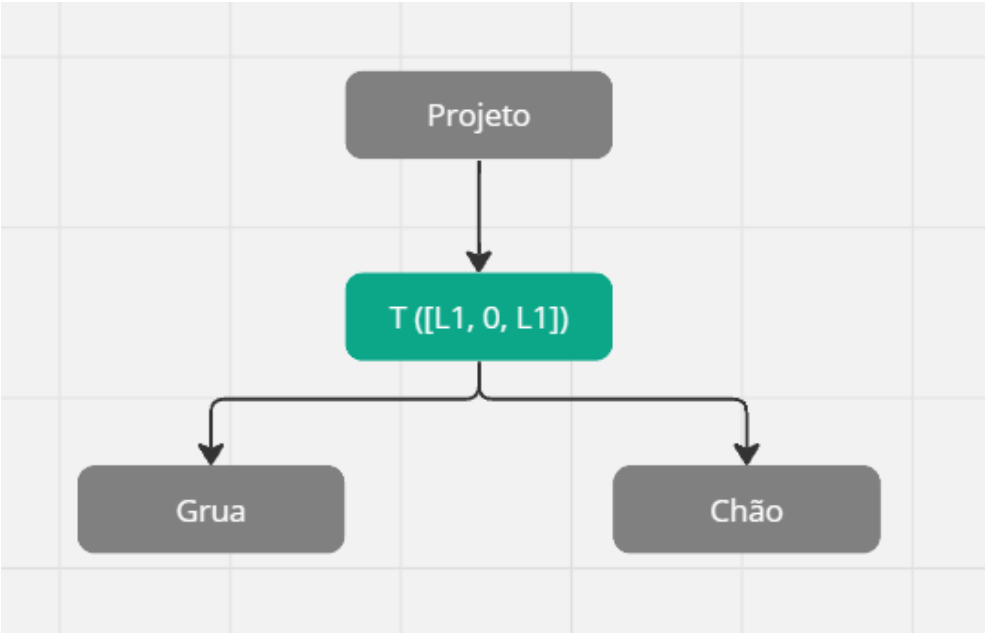
Projecto 2 - Modelação Hierárquica

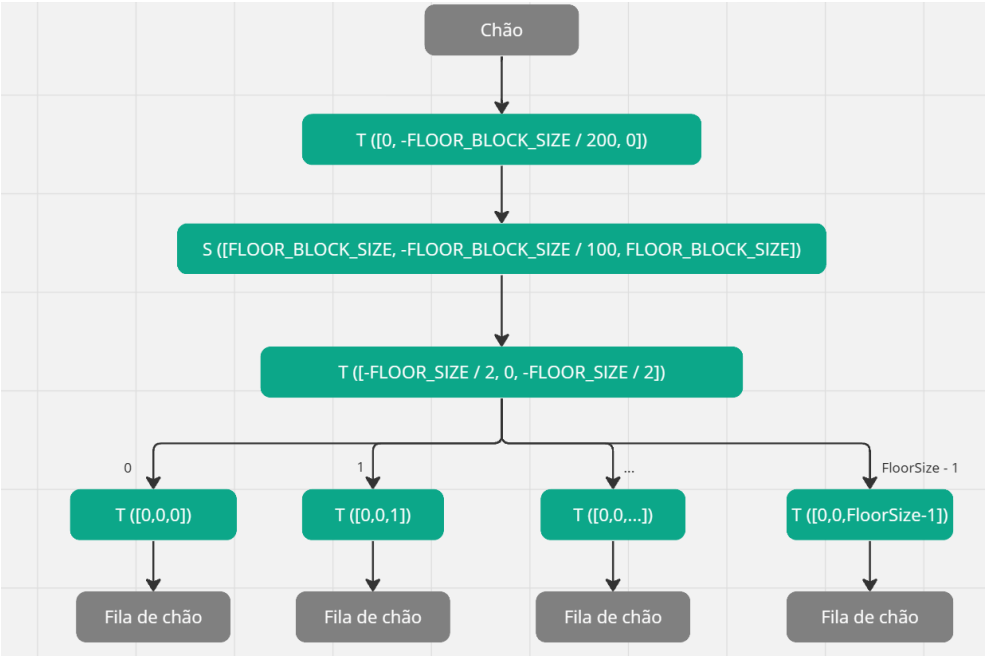


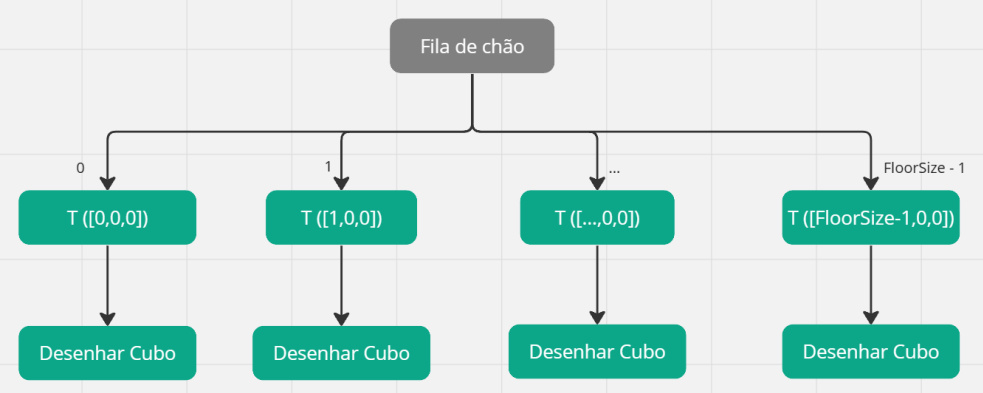
António Cinca Festas – 63739 – ac.festas@campus.fct.unl.pt

Gonçalo Carvalho – 61605 – gmm.carvalho@campus.fct.unl.pt

Grafo de Cena

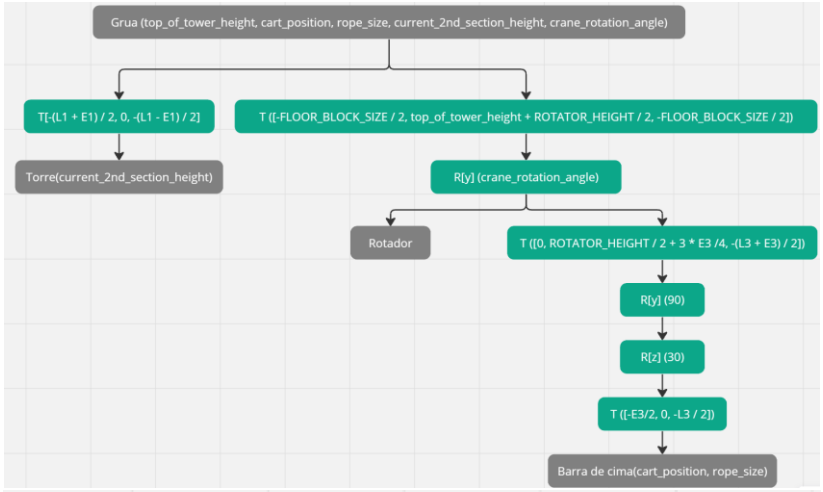


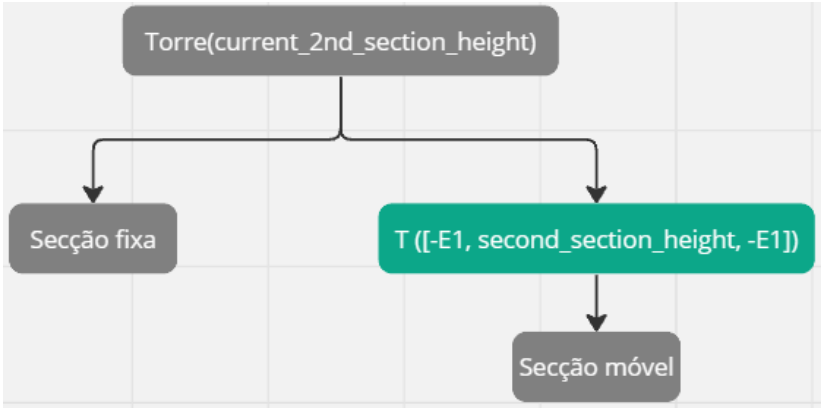


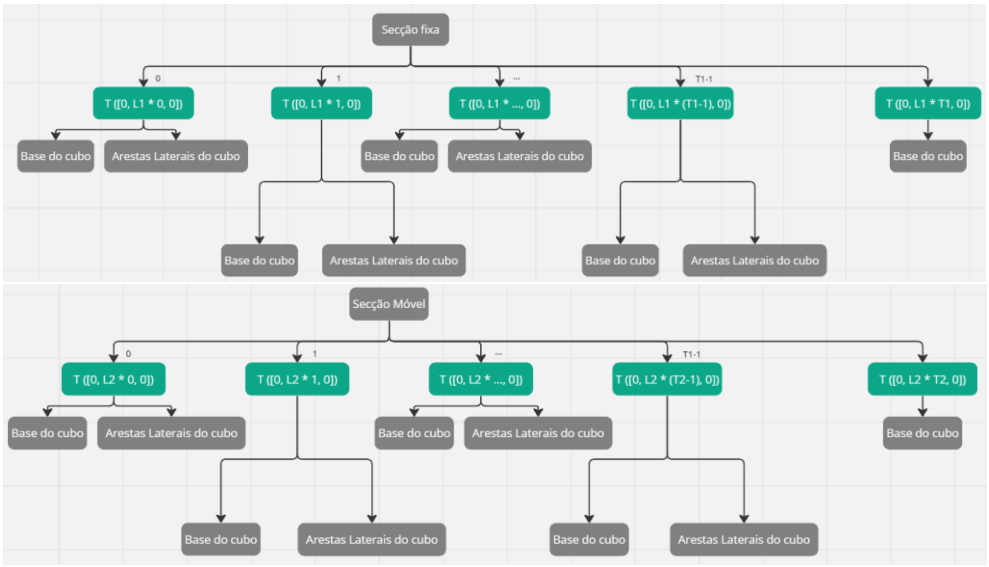


Nestas seguintes imagems a grua, e os seus componentes, recebem como argumentos as variáveis:

- top_of_tower_height – Altura actual do topo da torre.
- cart_position - Posição actual do carro deslizante.
- rope_size – Comprimento actual da corda do carro.
- current_2nd_section_height – Altura actual do chão até à secção móvel da torre.
- crane_rotation_angle – Ângulo de rotação da viga e do cilindro, a que chamámos de rotador.



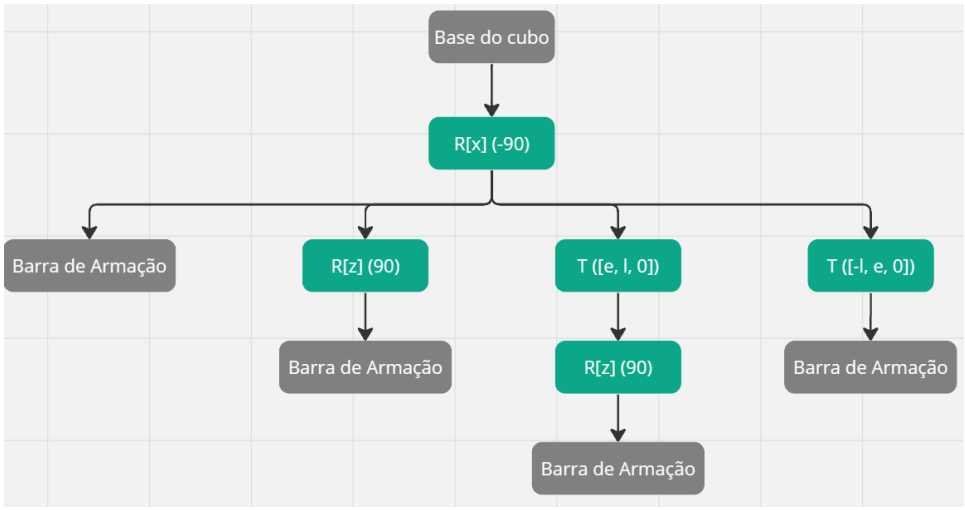


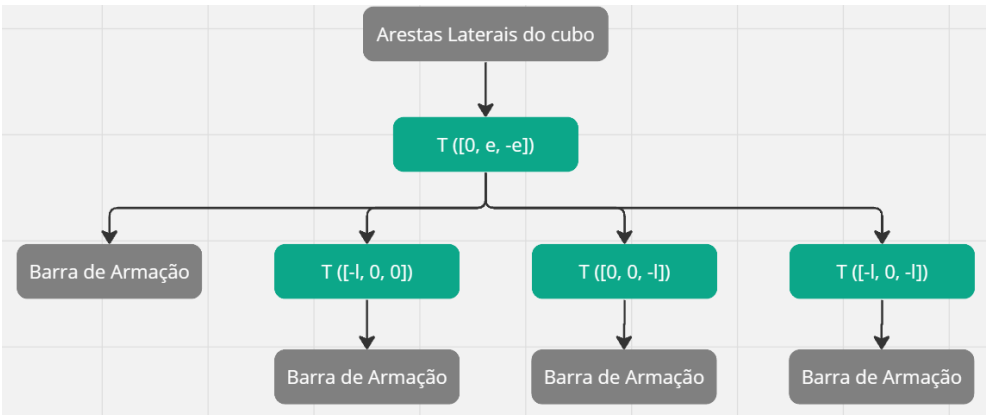


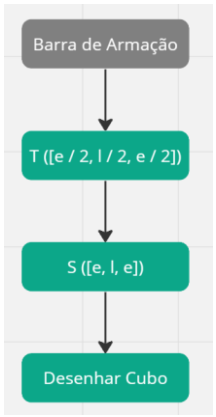
Para as funções: base do cubo, arestas laterais do cubo e barra de armação do cubo; as seguintes variáveis tomam valores diferentes dependendo de se as funções são chamadas pela secção móvel ou fixa.

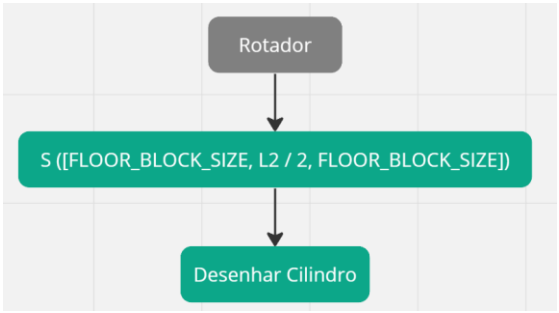
Se chamadas pela secção fixa: e = E1 e l = L1

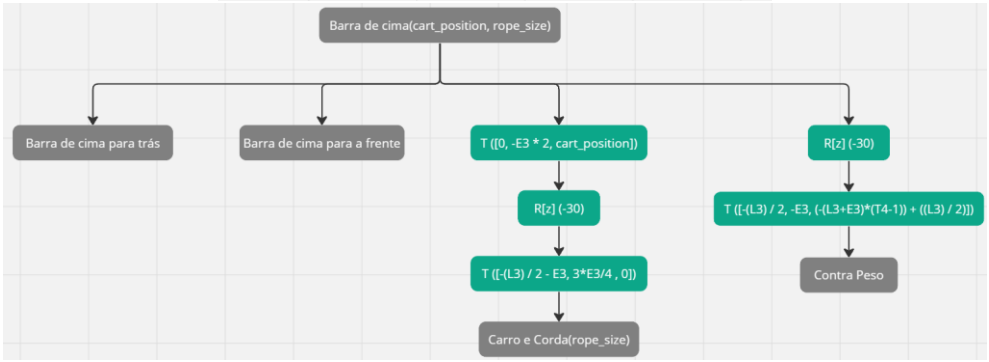
Se chamadas pela secção móvel: e = E2 e l = L2

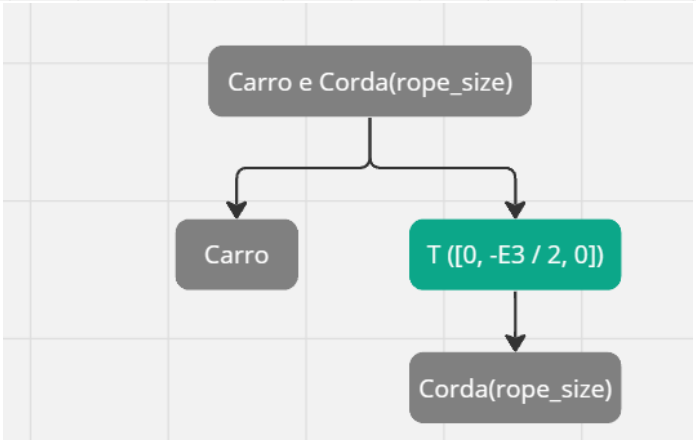


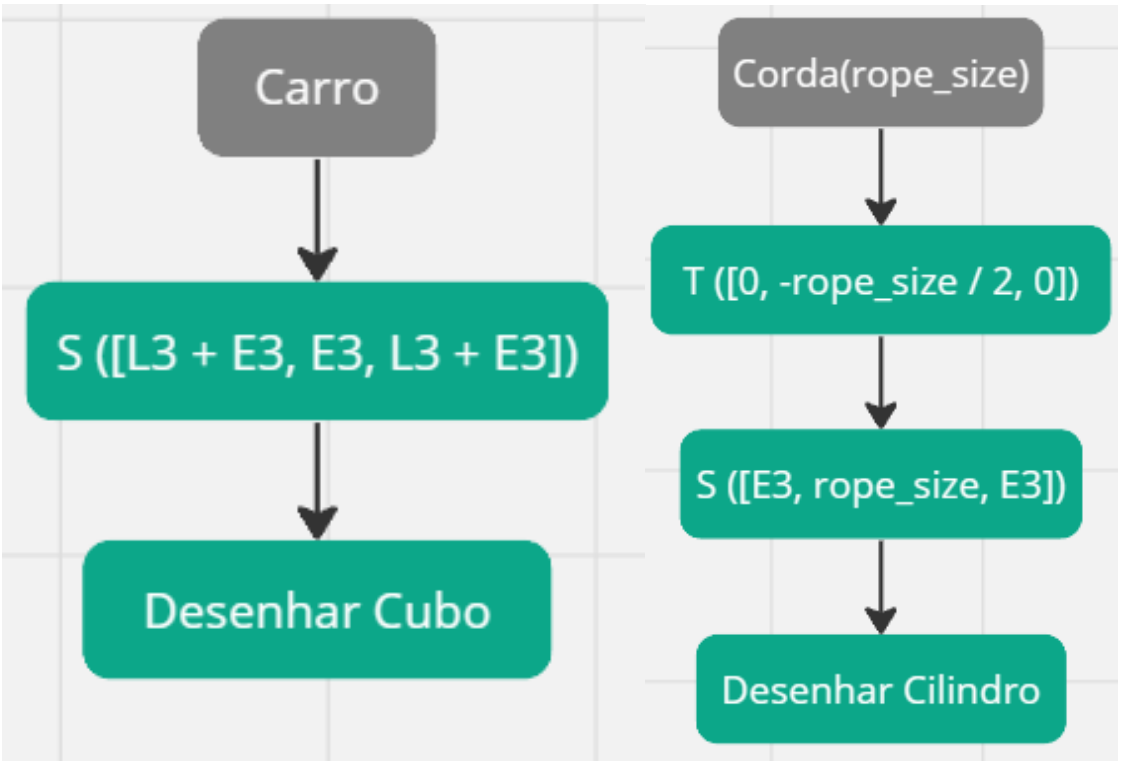

Arestas Laterais do cubo
T ([0, e, -e])
Barra de Armação
T ([-l, 0, 0])
T ([0, 0, -l])
T ([-l, 0, -l])
Barra de Armação
Barra de Armação
Barra de Armação

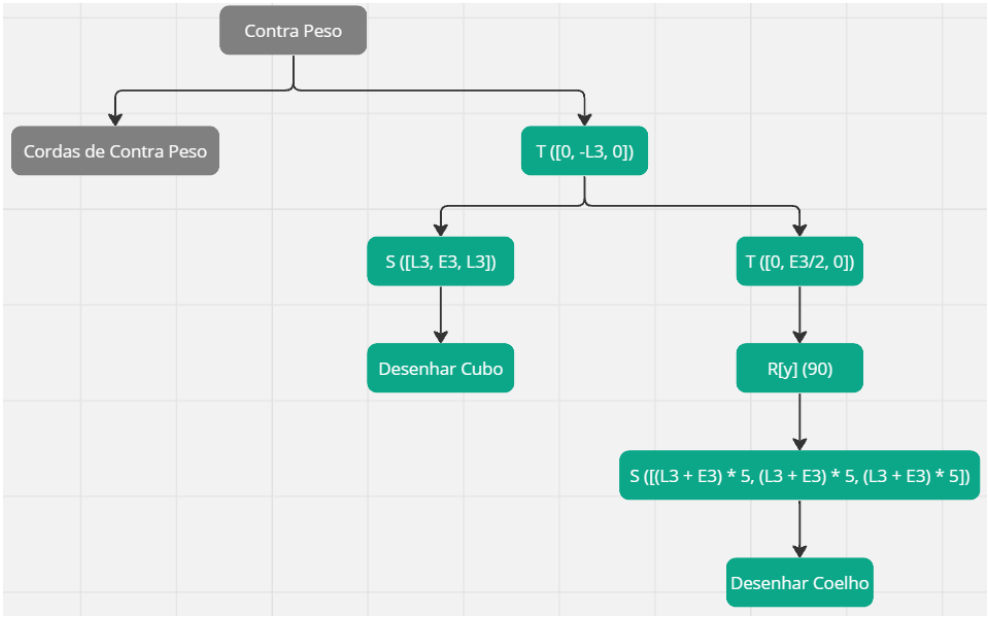

Barra de Armação
T ([e / 2, l / 2, e / 2])
S ([e, l, e])
Desenhar Cubo

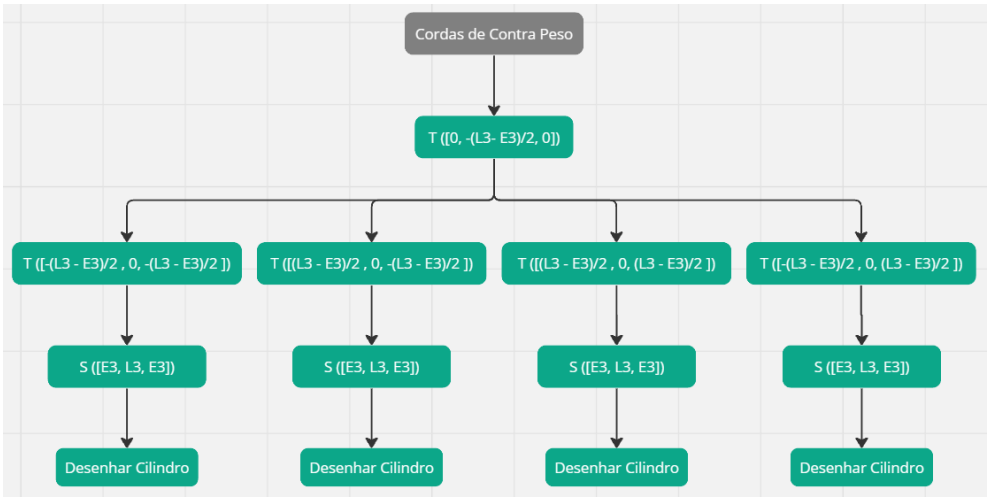

Rotador
S ([FLOOR_BLOCK_SIZE, L2 / 2, FLOOR_BLOCK_SIZE])
Desenhar Cilindro

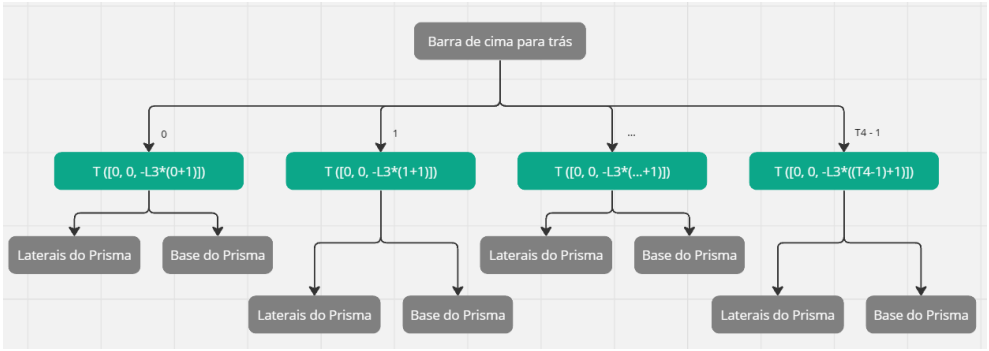

Barra de cima(cart_position, rope_size)
Barra de cima para trás
Barra de cima para a frente
T ([0, -E3 * 2, cart_position])
R[z] (-30)
T ([-(L3) / 2 - E3, 3*E3/4 , 0])
Carro e Corda(rope_size)
R[z] (-30)
T ([-(L3) / 2, -E3, (-(L3+E3)*(T4-1)) + ((L3) / 2)])
Contra Peso

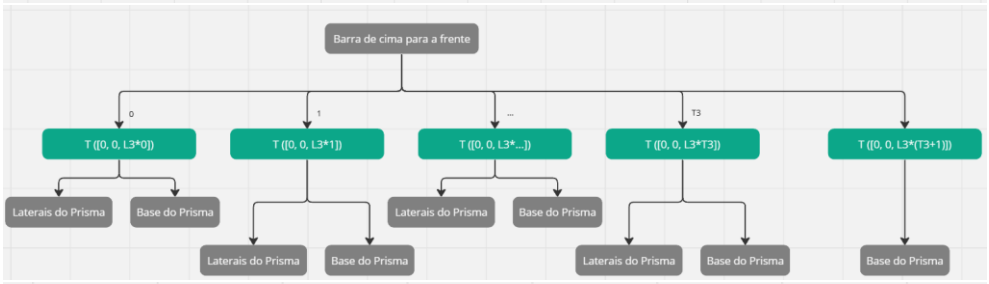

Carro e Corda(rope_size)
Carro
T ([0, -E3 / 2, 0])
Corda(rope_size)

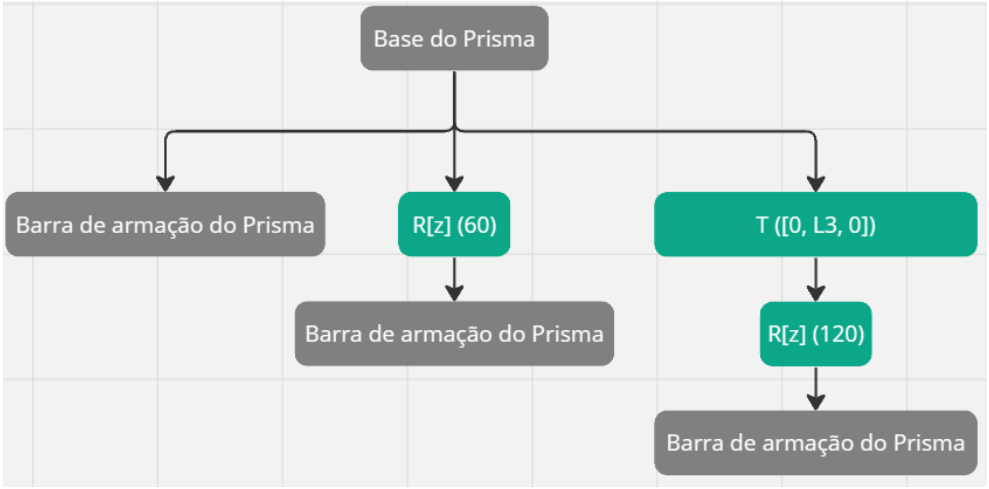

Carro
S ([L3 + E3, E3, L3 + E3])
Desenhar Cubo
Corda(rope_size)
T ([0, -rope_size / 2, 0])
S ([E3, rope_size, E3])
Desenhar Cilindro

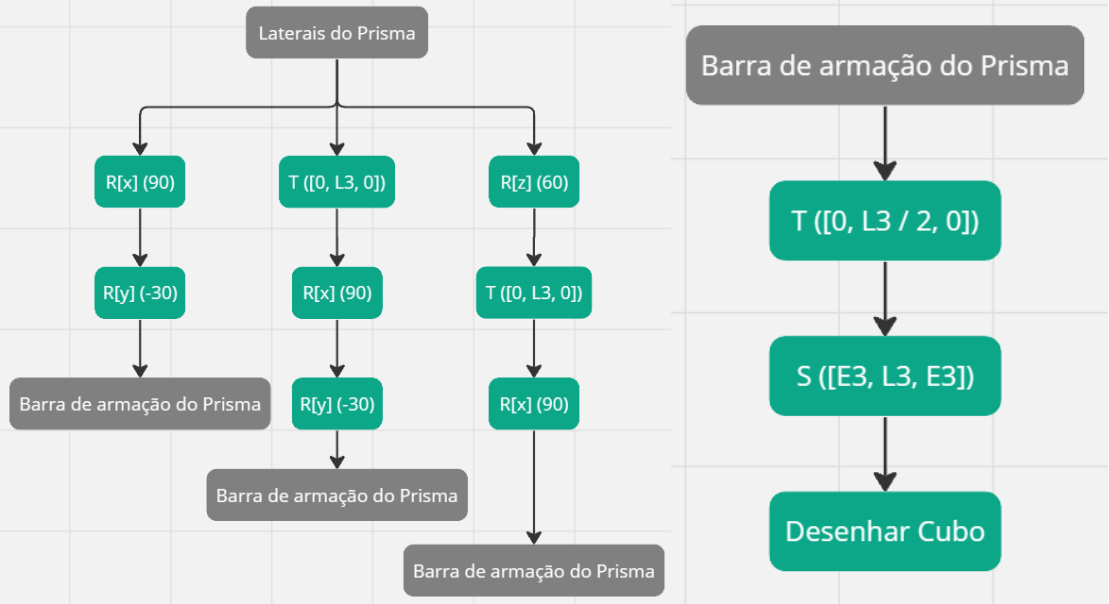

Contra Peso
Cordas de Contra Peso
T ([0, -L3, 0])
S ([L3, E3, L3])
Desenhar Cubo
T ([0, E3/2, 0])
R[y] (90)
S ([(L3 + E3) * 5, (L3 + E3) * 5, (L3 + E3) * 5])
Desenhar Coelho


Cordas de Contra Peso
T ([0, -(L3- E3)/2, 0])
T ([-(L3 - E3)/2 , 0, -(L3 - E3)/2])
T ([[(L3 - E3)/2 , 0, -(L3 - E3)/2])
T ([[(L3 - E3)/2 , 0, (L3 - E3)/2])
T ([-(L3 - E3)/2 , 0, (L3 - E3)/2])
S ([E3, L3, E3])
S ([E3, L3, E3])
S ([E3, L3, E3])
S ([E3, L3, E3])
Desenhar Cilindro
Desenhar Cilindro
Desenhar Cilindro
Desenhar Cilindro


Barra de cima para trás
0
1
...
T4 - 1
T ([0, 0, -L3*(0+1)])
T ([0, 0, -L3*(1+1)])
T ([0, 0, -L3*(...+1)])
T ([0, 0, -L3*((T4-1)+1)])
Laterais do Prisma
Base do Prisma
Laterais do Prisma
Base do Prisma
Laterais do Prisma
Base do Prisma
Laterais do Prisma
Base do Prisma


Barra de cima para a frente
0
1
...
T3
T ([0, 0, L3*0])
T ([0, 0, L3*1])
T ([0, 0, L3*...])
T ([0, 0, L3*T3])
T ([0, 0, L3*(T3+1)])
Laterais do Prisma
Base do Prisma
Laterais do Prisma
Base do Prisma
Laterais do Prisma
Base do Prisma
Laterais do Prisma
Base do Prisma
Base do Prisma


Base do Prisma
Barra de armação do Prisma
R[z] (60)
T ([0, L3, 0])
Barra de armação do Prisma
R[z] (120)
Barra de armação do Prisma


Laterais do Prisma
R[x] (90)
T ([0, L3, 0])
R[z] (60)
R[y] (-30)
R[x] (90)
T ([0, L3, 0])
Barra de armação do Prisma
R[y] (-30)
R[x] (90)
Barra de armação do Prisma
Barra de armação do Prisma
Barra de armação do Prisma
T ([0, L3 / 2, 0])
S ([E3, L3, E3])
Desenhar Cubo

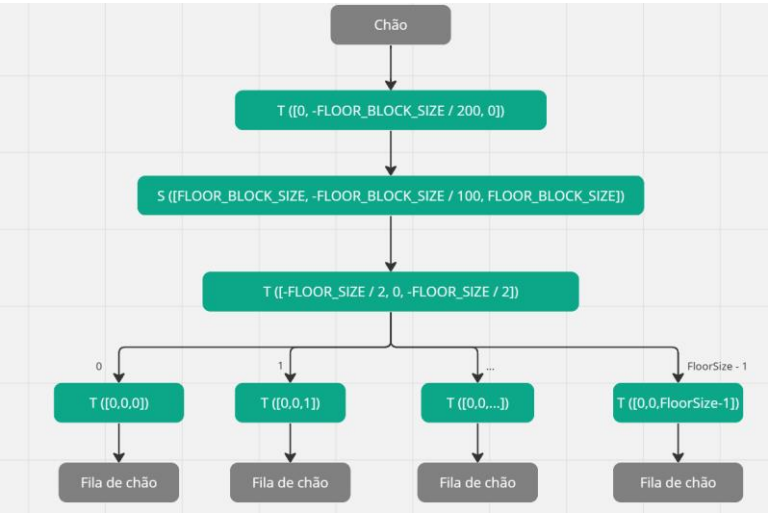
Extras

O contrapeso do guindaste consiste numa plataforma, segura por quatro cordas presas à viga da grua, e num coelho em cima dessa plataforma.

Explicação

Há “incoerências” entre o grafo de cena e o código, mas são propositadas.

Por exemplo, no segmento do grafo de cena sobre o chão:



E no código entregue:

```
for (let i = 0; i < FLOOR_SIZE; i++) { // Create each block of the floor
    pushMatrix();
    for (let j = 0; j < FLOOR_SIZE; j++) {
        let color = ((i + j) % 2) == 0 ? COLOR_FLOOR_1 : COLOR_FLOOR_2;
        uploadModelView(color);
        CUBE.draw(gl, program, mode);
        multTranslation([1, 0, 0]);
    }
    popMatrix();
    multTranslation([0, 0, 1]);
}

/**
 *      READ ME:
 *      Este cliclo for em comentário faz o mesmo que o anterior e é o representado no grafo de cena, mas é menos eficiente
 * pq faz mais push e pop de matrizes

for (let i = 0; i < FLOOR_SIZE; i++) { // Create each block of the floor
    pushMatrix();
        multTranslation([0, 0, i]);

        for (let j = 0; j < FLOOR_SIZE; j++) {
            pushMatrix();
                multTranslation([j, 0, 0]);
                let color = ((i + j) % 2) == 0 ? COLOR_FLOOR_1 : COLOR_FLOOR_2;

                uploadModelView(color);
                CUBE.draw(gl, program, mode);
            popMatrix();
        }
    popMatrix();

}
*/
```

O grafo entregue não corresponde ao código entregue, mas colocámos uma versão do código que está de acordo com o grafo em comentário.

Esta decisão foi tomada pois o grafo do código entregue seria estranho e complicado de mostrar e analisar. Mas mantemos a nossa decisão do código ficar diferente do grafo que apresentamos por razões de eficiência, o código comentado é menos eficiente porque realiza mais vezes pushMatrix(), popMatrix() e em alguns casos até multTranslation().