

A hand-drawn stick figure model of a person, used for biomechanical analysis. The figure is composed of several segments represented by ovals and circles, connected by joints marked with red dots. The joints are labeled with letters: 'n' for the head, 'h' for the neck, 'b' for the shoulder, 'c' for the elbow, 'd' for the wrist, 'e' for the hip, 'f' for the knee, 'g' for the ankle, 'h' for the foot, 'i' for the heel, 'j' for the ankle, 'k' for the foot, and 'l' for the heel. The dimensions are indicated by arrows and numbers: the head diameter is 1; the neck length is 2; the shoulder width is 1; the upper arm length is 3; the forearm length is 3; the hand length is 3; the torso length is 1; the hip width is 1; the thigh length is 3.5; the shank length is 2.5; the foot length is 0.5; and the foot width is 1.5. A coordinate system is shown in the top left corner with the y-axis pointing up and the x-axis pointing right.

```

graph TD
    a((a)) --> b((b))
    a --> c((c))
    b --> bc((c))
    b --> bm((m))
    b --> be((e))
    bc --> bcd((d))
    bm --> bmn((n))
    be --> bef((f))
    c --> cj((j))
    c --> ck((k))
    cj --> cjh((h))
    cjh --> cji((i))
    ck --> ckL((L))
  
```

$a \approx \vec{0}$

- vec3 origin
- vec3 scale
- vec3 position
- vec3 rotation (con respecto al origen)

 $\}$

3

DrawModel (M, view, Projection)



A hand-drawn diagram of an ellipse on a blue grid. A horizontal line segment from the center to the right edge is labeled 'a'. A vertical line segment from the center to the top edge is labeled 'b'.

Lock to position

A diagram showing a box labeled "Shader" with four arrows pointing to it from the right. The arrows are labeled "Model", "View", "Projection", and "Vec4 (color, 1F)".

Move point

```

Node {
    string name
    ellipsoid part
    vector node2 children
    bool endeffector
};

```

```

Model {
    Node head
};

```

Model puppet

```
puppet.translate("torso", somemove)
```

```
trans = mat4(vec3(somemove))
```

```
now = this->head
```

```
while (now.name == "torso" or now == endeffector)
```

```
    ↓ now.part.pushden(trans)
```

Repite esto mismo para rotate...

• Añadir bandera para restringir movimiento (en node).

- Translate:

- Rotate:

Ahora sob tengo que averiguar como hacer iluminacion y camera

☺

← Done owo

MatrixStack {

```
mat4 stack = I
```

```
stack operation_stack
```

```
push(A) {
```

```
    operation_stack.push(A)
```

```
    stack * = A
```

```
}
```

```
pop() {
```

```
    A = operation_stack.top();
```

```
    stack * = A-1
```

```
    operation_stack.pop();
```

```
}
```

}

Model push/pop

del libro (no o no)

unu

ESG es una mala idea



Sea $n :=$ "numero de primitivas del modulo" pero calculo inversa...

Complejidad de tiempo $O(n \log n)$

Complejidad de memoria $O(2n^2 \log n)$ too bad!

Mejor...

operation-stack $\rightarrow \prod_{i=0}^n A_i$ $n \leftarrow \text{top}$

$\Rightarrow \text{ops.push}(B) = B * S_n$

$\text{ops.pop}() = B^{-1} S_{n+1}$
 $= S_n$

$A_2 A_1 A_0$	2
$A_1 A_0$	1
A_0	0

Ejemp:

$$A_3 A_2 A_1 A_0 = S_n$$

$$B A_3 A_2 A_1 A_0 = S_{n+1}$$

$$\underbrace{B^{-1} B}_I A_3 A_2 A_1 A_0 = S_n$$

operation-stack {

vector os

push(A) {

mat $N = A * \text{os.top}()$

os.push(N)

}

pop() {

os.pop()

s

top() {

os.top

}

}

No hay inversa!

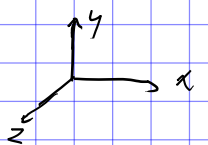
\rightarrow

en realidad
sob es un
wrapper de `std::stack`

Casi lo olvido...

Un objeto tiene

Atributos de su frame



Position (origin position)

orientation
scale

y Atributos respecto a su frame...



position
scale
rotation

Todos los atributos de frame son dependientes del padre

Entonces, sea $R(\theta, n)$, $S(n)$, $P(n)$ los generadores de matrices afines de rotación, escala y traslación, la matriz de transformación de frame es,

$$A = P(n_f) R(\theta, n_f) S(n)$$

Luego, sea \bar{O}^T la matriz de frame del padre, entonces la matriz del frame del objeto

$$\bar{a}^T = \bar{O}^T A \dots$$

Después se calcula B idéntica a A para transformaciones locales y

$$\text{mat Model} = \bar{a}^T B$$

Al siguiente nivel se pasa \bar{a}^T y repite...

Pero ahora esto:

push(A) {

mat N = A * os.top()

os.push(N)

}

Pasa a esto

push(A) {

mat N = os.top() * A

os.push(N)

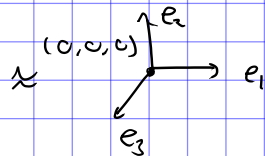
}

Y como que las matrices frame son del tipo

$$\bar{O}^T = \bar{\omega}^T O c$$

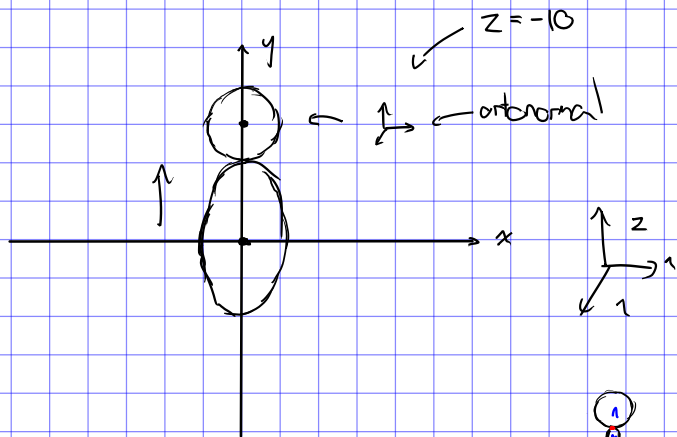
Ademas

$$\bar{C}^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



sud. tergo in major plan

Test



Puppet Render test :

```
Drawing "rightForearm"  
Drawing "torso"  
Drawing "hips"  
Drawing "leftUpperLeg"  
Drawing "leftLowerLeg"  
Drawing "leftFoot"  
Drawing "rightUpperLeg"  
Drawing "rightLowerLeg"  
Drawing "rightFoot"  
Drawing "shoulders"  
Drawing "neck"  
Drawing "head"  
Drawing "leftArm"  
Drawing "leftForearm"  
Drawing "rightArm"  
Drawing "rightForearm"
```

