Muelles

R. Fernandez
L. García
M. Gómez
D. Alejo

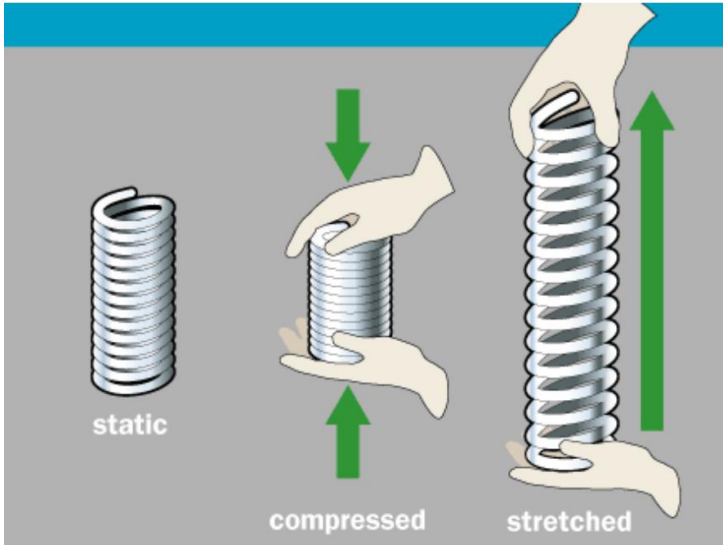
¿Qué es un muelle?

Muelle: definición

- Elemento elástico capaz de almacenar energía y desprenderse de ella sin sufrir deformación permanente cuando cesan las fuerzas o la tensión a las que es sometido.

Muelle: definición

- Tiene una posición de reposo
  - Está en la posición de reposo si no actúa ninguna fuerza
- Cuando actúa una fuerza sobre él
  - Se estira o se comprime
  - Ofrece una resistencia al cambio de posición

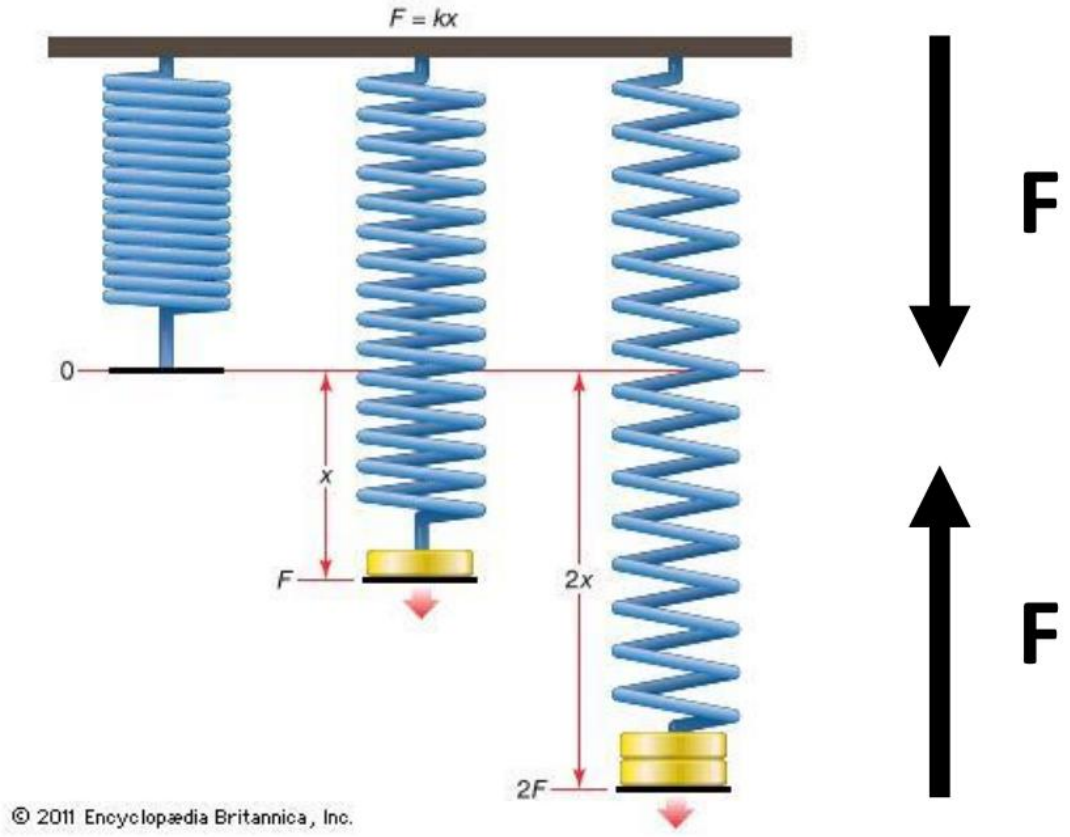


Muelle: definición

Física de un muelle

Ley de Hooke

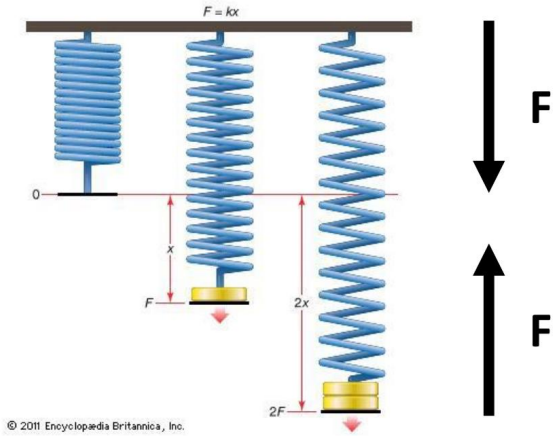
$$\vec{F} = -k \cdot \vec{x}$$



Ley de Hooke

- Constante elástica k (mide la rigidez del muelle).
- Cuanto mayor sea k más resistencia presenta el muelle a la deformación, su fuerza es mayor y por lo tanto más energía es capaz de almacenar.

$$\vec{F} = -k \cdot \vec{x}$$

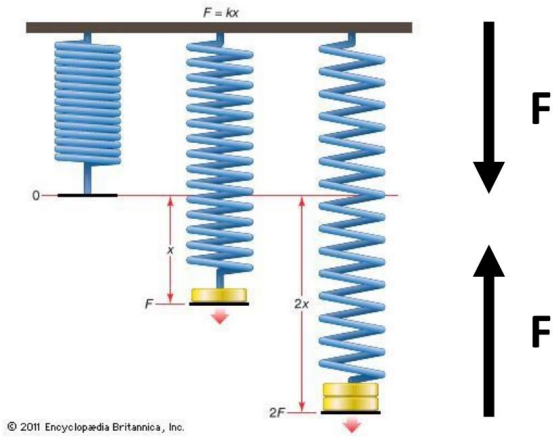


Ley de Hooke

$$\vec{F} = -k(|\vec{d}| - l_0) \cdot \left(\frac{\vec{d}}{|\vec{d}|}\right)$$

Cálculo vectorial de la fuerza.

- $\vec{d}$ Vector que une los dos extremos del muelle
- l_0 Longitud del muelle en reposo
- $|\vec{d}|$ Longitud actual del muelle (módulo)
- $(|\vec{d}| - l_0)$ Longitud deformada
- $\left(\frac{\vec{d}}{|\vec{d}|}\right)$ Dirección del muelle (Vector normalizado $\vec{d}$)



Compresión y extensión

- No todos los elementos son capaces de comprimirse y extenderse. Algunos solo aguantan un tipo de deformación.

Sólo extensión

Principalmente compresión

Límite de deformación

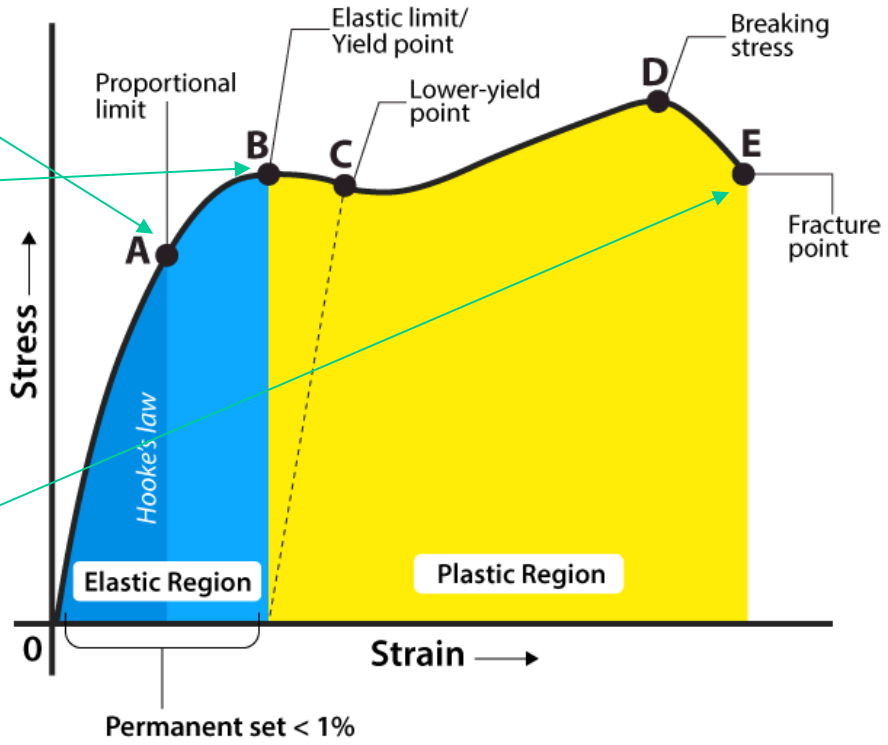
- Esta ley es una aproximación y solo es válida en una región pequeña de deformación. La gráfica muestra la fuerza que ejerce el muelle (stress) en función de la fuerza ejercida sobre el mismo (strain).

- Ley de Hook solo válida hasta A.

- Cuando ejercemos una fuerza o deformación mayor que B, el muelle empieza a perder la capacidad de revertir la deformación. Esto se conoce como ***límite elástico*** del material.

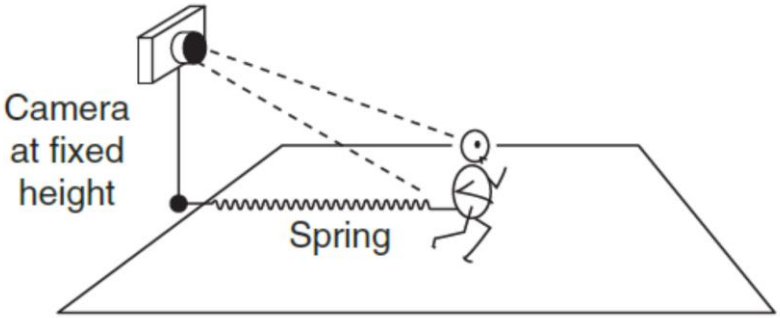
- Para fuerzas mayores que E**,** el muelle se rompe.

- También válido para la compresión, si tiene margen de deformación suficiente.

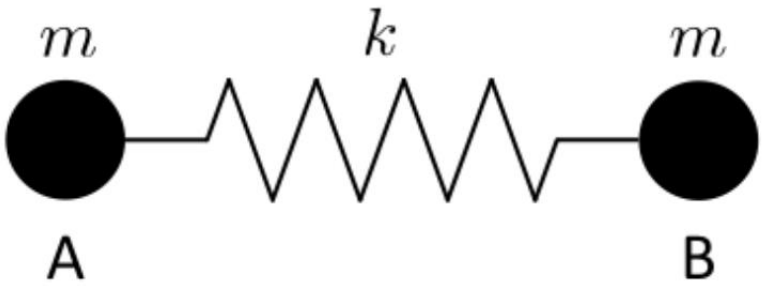


Aplicaciones

- Ley de elasticidad es muy general
- Con una partícula y un muelle podemos modelar casi cualquier cosa.
- Gomas elásticas, cuerdas del ring…
- Objetos flotando en el agua
- Cámaras siguiendo a personajes (Efecto más cinemático: Ejemplo)



- Usaremos los generadores de fuerza
- Tendremos que modelar los parámetros del muelle
- Implementar el código de la fuerza a aplicar
- Ejercicio 2 prácticas: Uniremos dos partículas A y B mediante un muelle
  - A sufrirá una fuerza debido al muelle conectado a A
  - B sufrirá una fuerza debido al muelle conectado a B
  - Necesitamos un par de generadores por muelle



Código C++: Class ParticleSpring

```
#pragma once

#include "ForceGenerator.h"
#include "core.hpp"

class SpringForceGenerator :public ForceGenerator {
public:
    SpringForceGenerator(double k, double resting_length, Particle *other);

    virtual void updateForce(Particle* particle);

    inline void setK(double k) { _k = k; }

    virtual ~SpringForceGenerator() {}
protected:
    double _k; // Elastic Coeff.
    double _resting_length;
    Particle* _other;
};
```

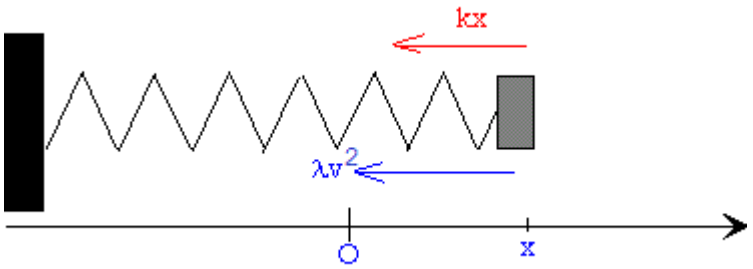
Código C++: updateForce

```cpp
#include "SpringForceGenerator.h"

SpringForceGenerator::SpringForceGenerator(double k, double resting_length, Particle* other) {
    _k = k;
    _resting_length = resting_length;
    _other = other;
}

void SpringForceGenerator::updateForce(Particle* particle) {
    // Particle is the particle to apply the force
    Vector3 relative_pos_vector = _other->getPos() - particle->getPos();
    Vector3 force;

    // normalize: Normalize the relative_pos_vector and returns its length.
    const float length = relative_pos_vector.normalize();
    const float delta_x = length - _resting_length;

    force = relative_pos_vector * delta_x * _k;

    particle->addForce(force);
}
```

Uso del generador

```
void ParticleSystem::generateSpringDemo() {
    // First one standard spring uniting 2 particles
    Particle* p1 = new Particle({ -10.0,10.0,0.0 }, { 0.0,0.0,0.0 }, { 0.0,0.0,0.0 }, 0.85, 60);
    Particle* p2 = new Particle({ 10.0,10.0,0.0 }, { 0.0,0.0,0.0 }, { 0.0,0.0,0.0 }, 0.85, 60);
    p2->setMass(2.0);
    SpringForceGenerator *f1 = new SpringForceGenerator(1, 10, p2);
    _force_registry.addRegistry(f1, p1);
    SpringForceGenerator *f2 = new SpringForceGenerator(1, 10, p1);
    _force_registry.addRegistry(f2, p2);
    _force_generators.push_back(f1);
    _force_generators.push_back(f2);
    _particles.push_back(p1);
    _particles.push_back(p2);
```

- Una partícula unida a otra partícula nos permite definir gran parte de las interacciones que necesitamos.
- No siempre usamos un muelle para unir dos partículas
- Una partícula unida a una posición fija
- Necesitamos otro sistema
  - ¿Cómo?



Código C++: Class ParticleAnchoredSpring

AnchoredSpringFG.h

```h
#pragma once
#include "SpringForceGenerator.h"
#include "core.hpp"

class AnchoredSpringFG :public SpringForceGenerator {
public:
    AnchoredSpringFG(double k, double resting, const Vector3& anchor_pos);

    ~AnchoredSpringFG();
};
```

AnchoredSpringFG.cpp

```cpp
#include "AnchoredSpringFG.h"

AnchoredSpringFG::AnchoredSpringFG(double k, double resting, const Vector3& anchor_pos) :
    SpringForceGenerator(k, resting, nullptr) {
    _other = new Particle(anchor_pos, { 0,0,0 }, { 0,0,0 }, 0, 1e6, 0.0, BOX);


}

AnchoredSpringFG::~AnchoredSpringFG() {
    delete _other;
}
```

Uso del generador

ParticleSystem.cpp

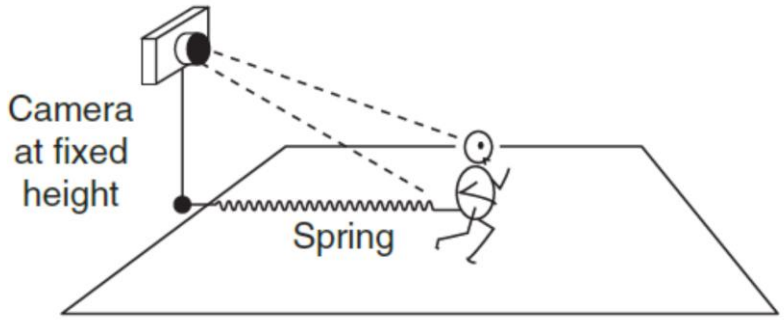
```
void ParticleSystem::generateSpringDemo() {
    // First one standard spring uniting 2 particles
    Particle* p1 = new Particle({ -10.0,10.0,0.0 }, { 0.0,0.0,0.0 }, { 0.0,0.0,0.0 }, 0.85, 60);
    Particle* p2 = new Particle({ 10.0,10.0,0.0 }, { 0.0,0.0,0.0 }, { 0.0,0.0,0.0 }, 0.85, 60);
    p2->setMass(2.0);
    SpringForceGenerator *f1 = new SpringForceGenerator(1, 10, p2);
    _force_registry.addRegistry(f1, p1);
    SpringForceGenerator *f2 = new SpringForceGenerator(1, 10, p1);
    _force_registry.addRegistry(f2, p2);
    _force_generators.push_back(f1);
    _force_generators.push_back(f2);
    _particles.push_back(p1);
    _particles.push_back(p2);

    // Then one spring with one fixed side
    Particle* p3 = new Particle({ -10.0,20.0,0.0 }, { 0.0,0.0,0.0 }, { 0.0,0.0,0.0 }, 0.85, 60);
    AnchoredSpringFG* f3 = new AnchoredSpringFG(1, 10, { 10.0,20.0,0.0 });
    _force_registry.addRegistry(f3, p3);
    _force_generators.push_back(f3);
    _particles.push_back(p3);
}
```

Ejemplos de uso: Goma elástica

- Sólo ejerce fuerza cuando se extiende
- Útil cuando queremos mantener dos partículas juntas que:
  - Pueden estar todo lo cerca que quieran
  - No pueden separarse más de una determinada distancia

Ejemplos de uso: Cámara dinámica

- La posición del personaje sería nuestro anclaje
- A la posición del personaje NO le afecta la fuerza del muelle
  - No hay que buscar el generador que actualice la posición
- La posición de la cámara SÍ se ve afectada por la fuerza del generador.



Ejemplos de uso: Fuerza de flotación

- Si queremos modelar cómo flota un objeto hay que gestionar las fuerzas implicadas
- El peso del objeto empuja el objeto hacia abajo.
- La fuerza de flotación empuja al objeto hacia arriba. El líquido desplazado ejerce una fuerza igual a su peso al elemento hundido.
  - **Peso** [N] = **m** [Kg]***g**[m/s^2] (F=m*a)
  - **m** [Kg] = **Vd** [L] * **d** [Kg/L]
- Principio Arquímedes: Volumen sumergido (Vs) = Volumen desplazado (Vd)
  - **Vd** [m^3] = **Vs** [m^3] = **S** [m^2] * **h** [m] (Si elemento uniforme S=cte)

$$F = E = d \cdot g \cdot V_s = \underbrace{d \cdot g \cdot S}_{k} \cdot x = dV\frac{x}{h}$$

Siendo *E* la fuerza de flotación, *d* la densidad del agua, *S* la sección (área) del objeto sumergido, *g* la constante de gravedad, *h* la altura del objeto sumergido y *m* la masa del liquido desplazado.

Fuerza de flotación

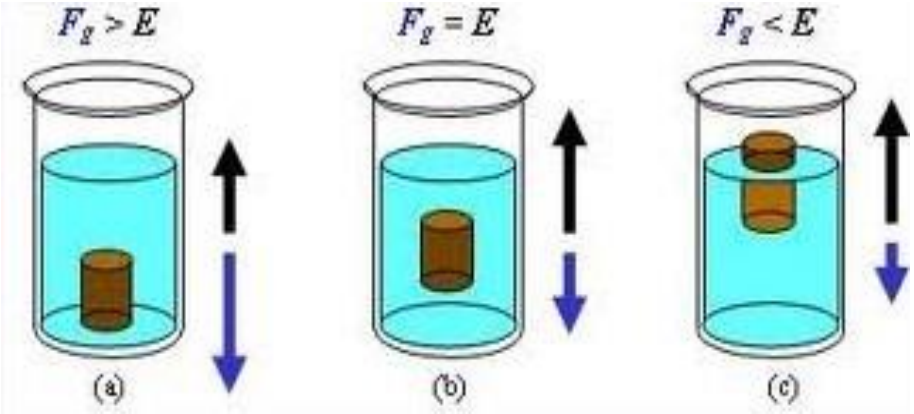
- Para calcular el empuje en general necesitamos saber la forma del objeto para obtener el volumen inmerso del objeto.
- Hay una profundidad máxima
  - Cuando el objeto está sumergido del todo ya no aumenta el empuje por mucho que se sumerja más
- En nuestro caso simplificamos suponiendo un objeto con forma uniforme (Sección Constante)
- El empuje es linealmente proporcional a la longitud que está hundido el objeto (d, g y S constantes)

$$F = d \cdot g \cdot S \cdot x = k * x$$

- Modelamos el empuje del líquido como un muelle

¿Flota o no flota?

- Flota o no flota en función de la densidad del líquido y el objeto:



$F_g = d_o V_o g$ $\quad$ $E = d_l V_o g$

Hundimiento : $d_o > d_l$

Suspensión: $d_o = d_l$

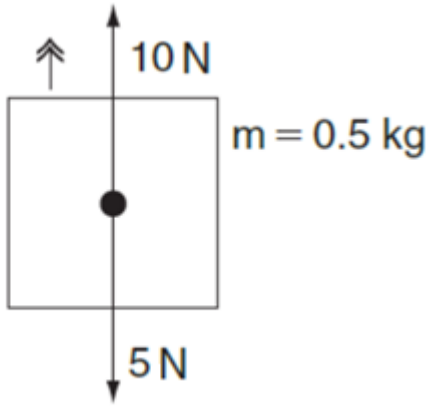
Flotación : $d_o < d_l$

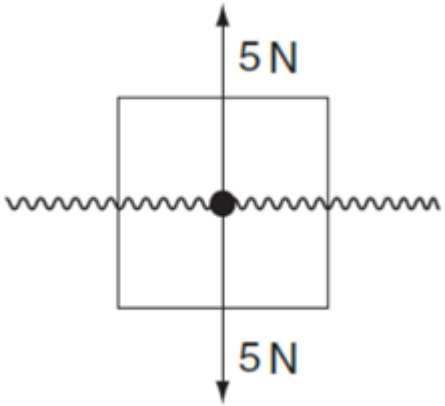
Fuerza de flotación

- Un cubo de 0.5Kg de masa y de 0.001m³ de volumer
- Peso del cubo: 0,5Kg*10m/s^2= 5 N (Kg.m/s^2)
- Inmersión total: 0.001m³=1dm³ de agua → 1Kg (depende de la densidad del líquido)
- Inmersión total: Peso del agua desalojada 10 N
- ¿Cuándo se estabilizará el cubo?
  - Cuando se alcance el equilibrio:

    Peso agua desalojada = Peso cubo

  Es decir, cuando esté hundido la mitad del objeto.



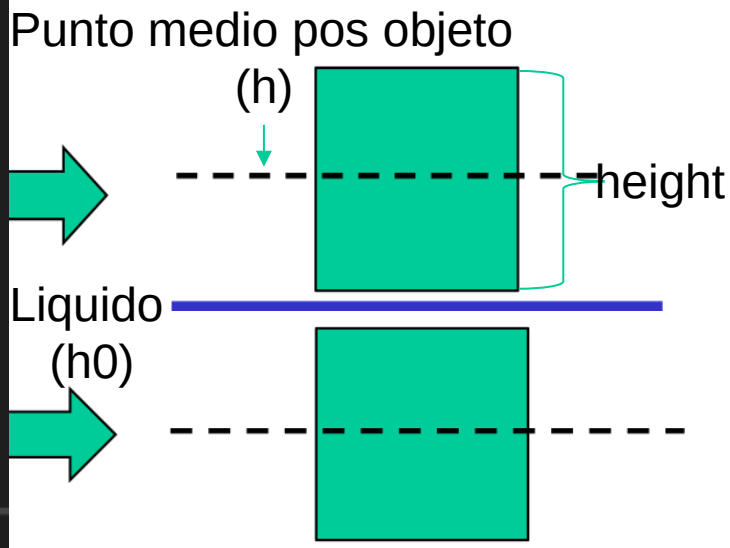


Código C++: class BuoyancyForceGenerator

```cpp
#pragma once

#include "ForceGenerator.h"
#include "core.hpp"

class BuoyancyForceGenerator :public ForceGenerator {
public:
    BuoyancyForceGenerator(float h, float V, float d);

    virtual void updateForce(Particle* particle);

    virtual ~BuoyancyForceGenerator();
protected:
    float _height;
    float _volume;
    float _liquid_density;
    float _gravity = 9.8;

    Particle* _liquid_particle; // For representation
};
```

Código C++: updateForce

```cpp
void BuoyancyForceGenerator::updateForce(Particle* p) {
    float h = p->getPos().y;
    float h0 = _liquid_particle->getPos().y;

    Vector3 f(0, 0, 0);
    float immersed = 0.0;
    if (h - h0 > _height * 0.5 ) {
        immersed = 0.0;
    } else if (h0 - h > _height * 0.5 ) {
        // Totally immersed
        immersed = 1.0;
    } else {
        immersed = (h0 - h) / _height + 0.5;
    }
    f.y = _liquid_density * _volume * immersed * 9.8;

    p->addForce(f);
}
```



Profundidad normalizada

h: Posición punto medio del objeto; h0 = posición superficie agua.

- En un muelle normal este sistema funciona bien
- El problema es si aumentamos la rigidez del muelle (k muy grande)
- Las fuerzas aplicadas aumentan
- El sistema oscila hasta el infinito (Fuerzas muy bruscas)
- No se puede gestionar así.
  - Hay que usar restricciones (*constraints*)
  - Pero eso ya es otro tema…