Proyecto Final - SPH

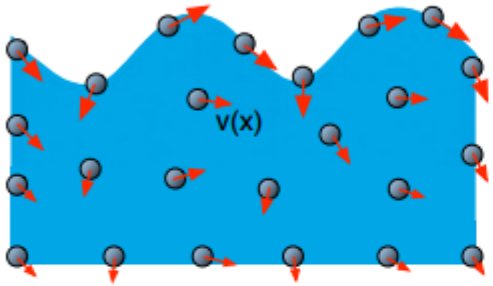
Jhonatan Core

INTRODUCCIÓN

* **Smoothed-particle hydrodynamics (SPH)** es un método computacional utilizado para simular fluidos. Fue creado por Gingold, Monaghan y Lucy (1977) inicialmente para problemas astrofísicos.

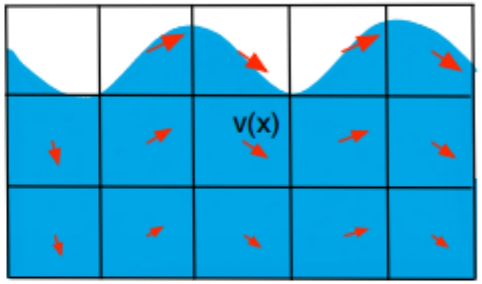
Lagrangiano vs Euleriano

Langrangian viewpoint



- Particles can move freely in space, carry quantities
- Fluid motion by moving particles

Eulerian viewpoint



- Fixed spatial locations
- Measure quantities as it flows past

Ecuaciones de Navier-Stokes

$$\vec{F}_i^{viscosity} = \frac{\mu}{\rho_i} \sum_j (\vec{u_j} - \vec{u_i}) m_j \nabla^2 W_{viscosity}(\vec{r_i} - \vec{r_j}, h)$$

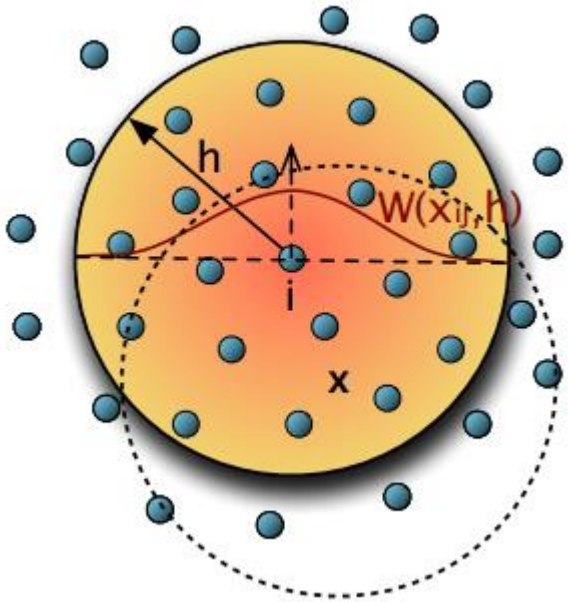
$$\vec{F}_i^{pressure} = -\sum_{i \neq j} \frac{p_i + p_j}{2} \frac{m_j}{\rho_j} \nabla W_{pressure}(\vec{r_i} - \vec{r_j}, h)$$

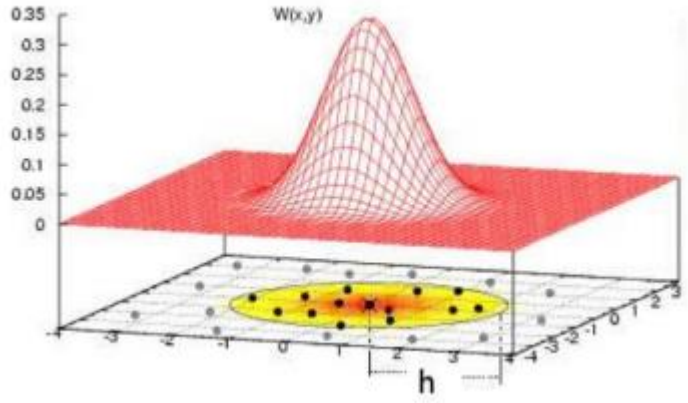
$$\vec{F}_i^{tension} = -\sigma \hat{n} \nabla^2 c_i$$

$$\nabla^2 c_i = \sum_j \frac{m_j}{\rho_j} \nabla^2 W_{default}(\vec{r_i} - \vec{r_j}, h)$$

Kernel

$$A(\mathbf{x}) = \sum_j \frac{m_j}{\rho_j} A_j W(\mathbf{x} - \mathbf{x_j}, h)$$





Bucle de simulación

```
1: for all particles i do
2:     apply forces v_i ⇐ v_i + Δt f_ext(x_i)
3:     predict position x_i* ⇐ x_i + Δt v_i
4: end for
5: for all particles i do
6:     find neighboring particles N_i(x_i*)
7: end for
8: while iter < solverIterations do
9:     for all particles i do
10:        calculate λ_i
11:    end for
12:    for all particles i do
13:        calculate Δp_i
14:        perform collision detection and response
15:    end for
16:    for all particles i do
17:        update position x_i* ⇐ x_i* + Δp_i
18:    end for
19: end while
20: for all particles i do
21:    update velocity v_i ⇐ (1/Δt)(x_i* − x_i)
22:    apply vorticity confinement and XSPH viscosity
23:    update position x_i ⇐ x_i*
24: end for
```

Paso 1 - Búsqueda de vecinos

Paso 2 - Cálculo de la densidad y presión

Paso 3 - Cálculo de las fuerzas externas, la presión y la viscosidad

Paso 4 - Actualización de velocidad y posición

Paso 5 - Cálculo de colisiones con el recipiente

Paso 6 - Renderizado

IMPLEMENTACIÓN

Paso 1 – Búsqueda de vecinos

* **sph_cell_init.glsl:** Se encarga de inicializar las celdas poniendo su posición en -1.

* **sph_grid.glsl:** Se encarga de incluir dentro del grid las partículas contenidas en cada bucket.

IMPLEMENTACIÓN

Paso 2 - Cálculo de la densidad y la presión

* **sph_density.glsl:** Se encarga del cálculo de la densidad y de la presión a la que está sometida una particula

```
uniform samplerBuffer sData0;  -> posición + densidad
uniform samplerBuffer sData1;  -> velocidad
```

IMPLEMENTACIÓN

Paso 3 - Cálculo de las fuerzas externas

Paso 4 - Actualización de velocidad y posición

Paso 5 - Cálculo de colisiones con el recipiente

* **sph_forces.glsl:** Se encargará de calcular las fuerzas a las que está sometida cada partícula con respecto a sus vecinas, actualizar la posición y la velocidad y posteriormente redistribuir la posición si la partícula se sale del recipiente.

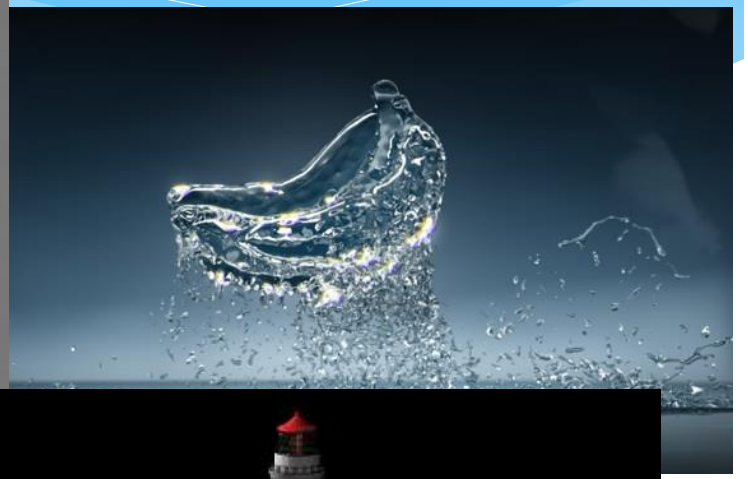
IMPLEMENTACIÓN

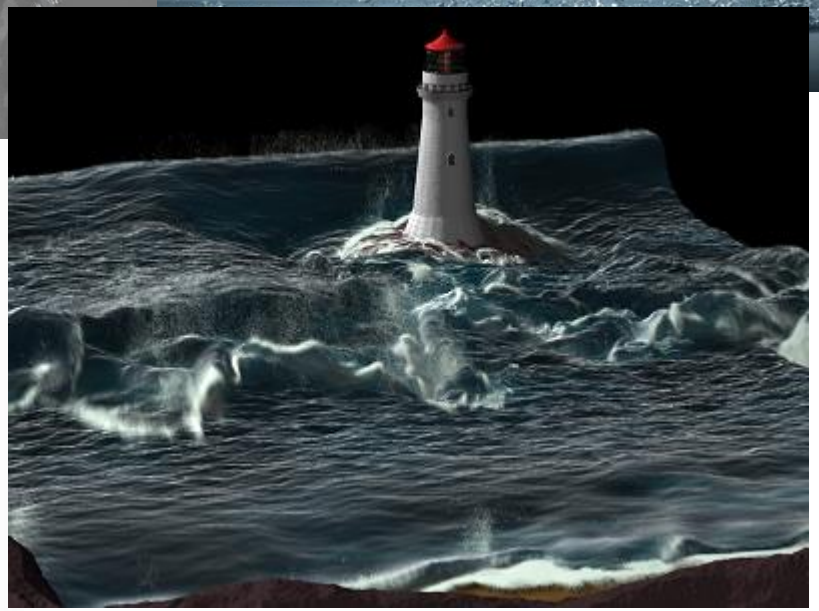
Paso 6 – Renderizado

* **cube.glsl:** Se utiliza para dibujar el cubo que contiene las partículas de nuestro sistema.
* **sph_bucket_render.glsl:** Se encarga de dibujar los buckets o celdas por las que está compuesta nuestro grid.
* **sph_render.glsl:** Se encarga de dibujar las partículas.

RESULTADO

Posibles resultados futuros





Gracias

Jhonatan Core