Approximate and exact buoyancy calculation for real time floating simulation of meshes

Gábor FÁBIÁN
ELTE Eötvös Loránd University, Budapest, Hungary

insomnia@inf.elte.hu

Motivation

Problem statement

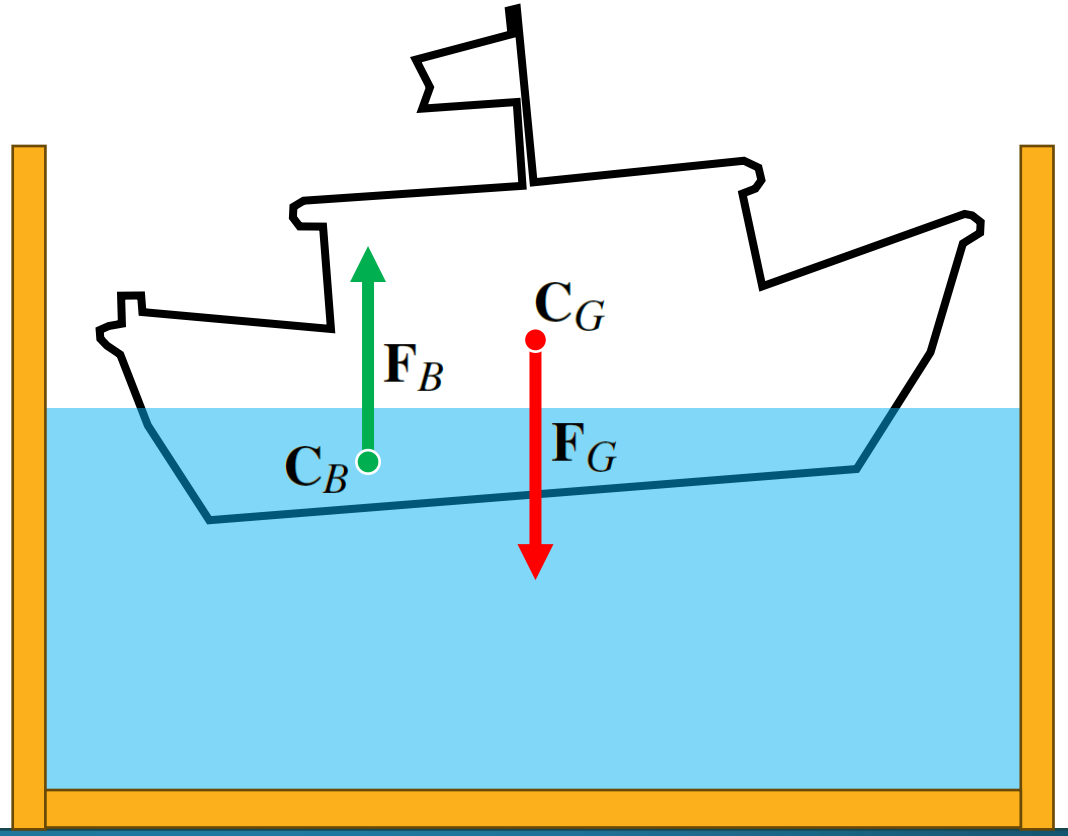
Floating = gravity + buoyancy

$$\mathbf{F}_G = -mg\mathbf{j}$$

$$\mathbf{F}_B = \rho g V_B \mathbf{j}$$

Archimedes' principle

Submerged volume



Volume and centroid of meshes

$$\mathrm{vol}(\Omega) = \iiint_{\Omega} 1 \, \mathrm{d}V = \frac{1}{3} \oiint_{\partial\Omega} \mathbf{id} \cdot \mathrm{d}\mathbf{S}$$

Integrate over a **closed** surface

Sum of triangle contributions

$$V = \sum_i V_i(\mathbf{o})$$

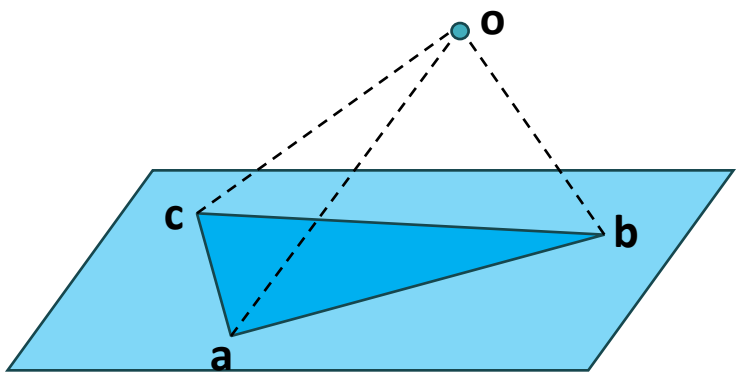
$$V_i(\mathbf{o}) := \frac{1}{6}(\mathbf{a}_i - \mathbf{o}) \cdot \Big((\mathbf{b}_i - \mathbf{o}) \times (\mathbf{c}_i - \mathbf{o}) \Big)$$

Volume element

$$\mathbf{C} = \frac{1}{V} \sum_i V_i(\mathbf{o}) \mathbf{C}_i(\mathbf{o})$$

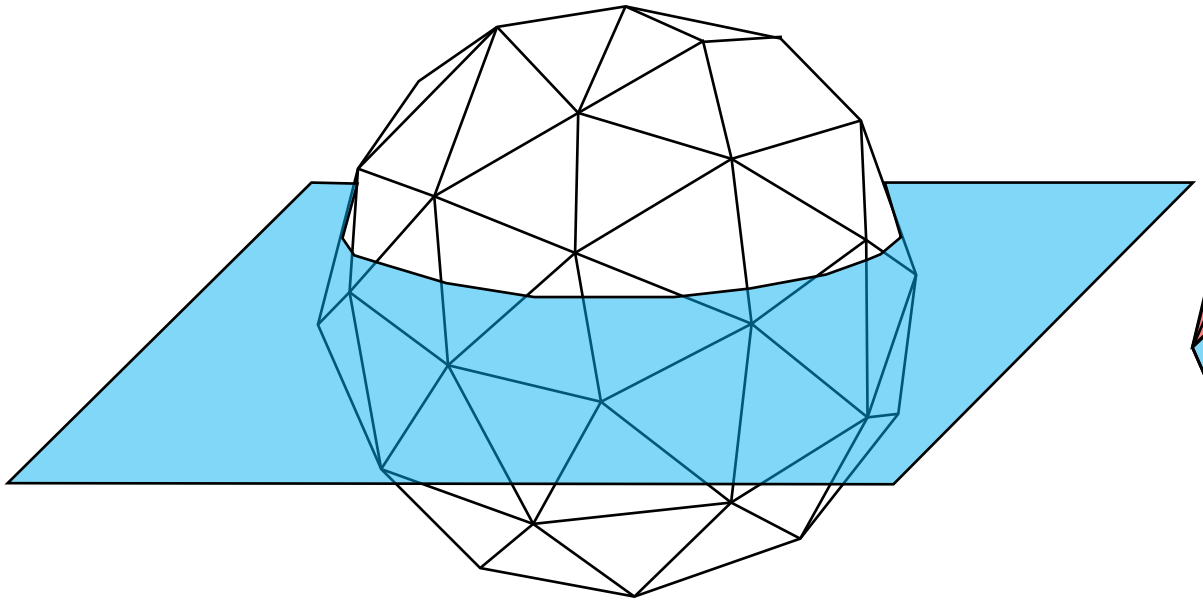
$$\mathbf{C}_i(\mathbf{o}) = \frac{\mathbf{a}_i + \mathbf{b}_i + \mathbf{c}_i + \mathbf{o}}{4}$$

Centroid element



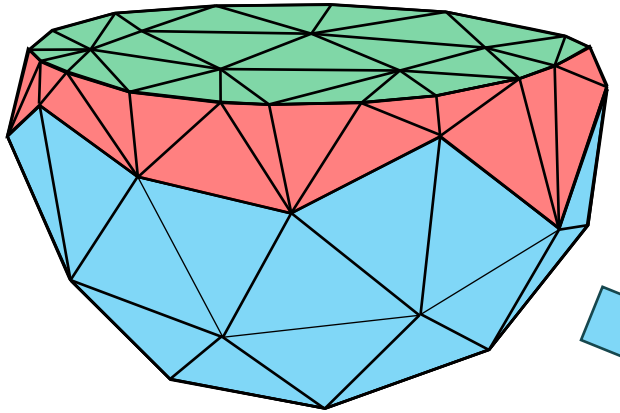
V≠0

The computational problem



Integrate over a **closed** surface

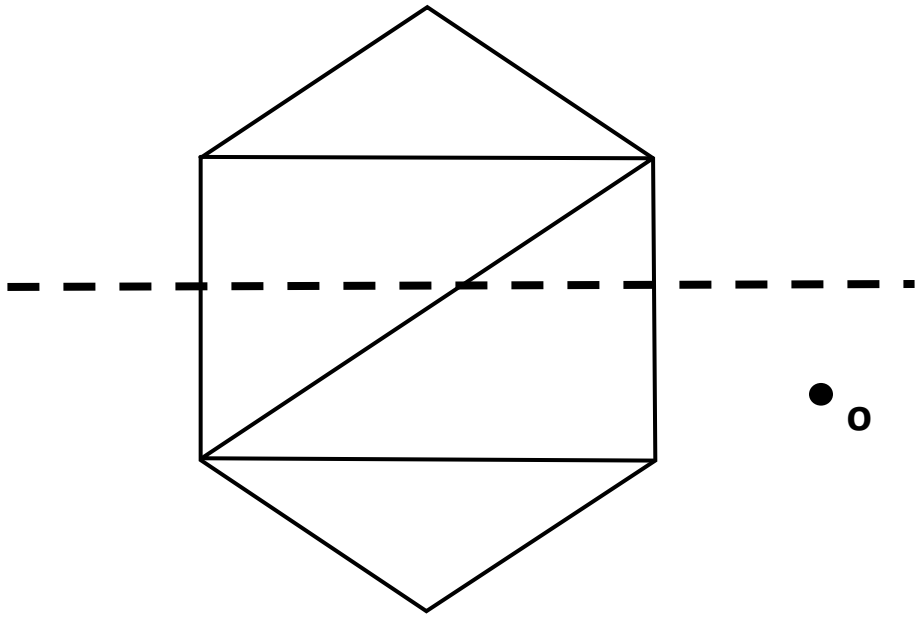
ON the surface ???



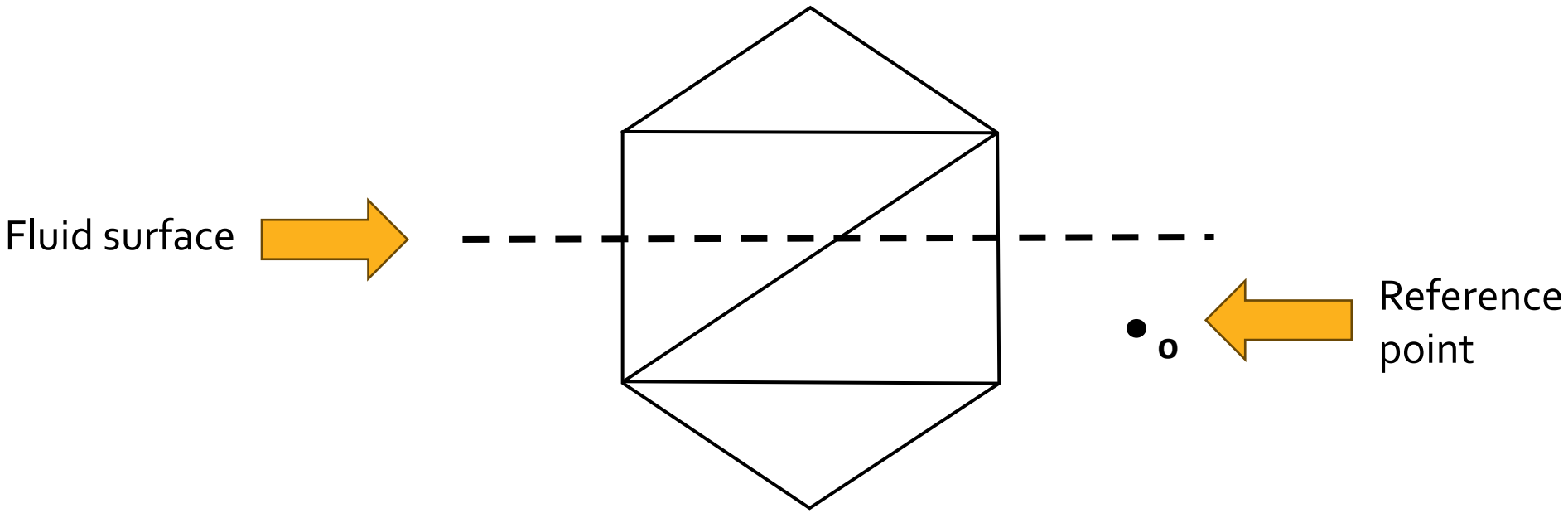
Partially submerged

Fully submerged

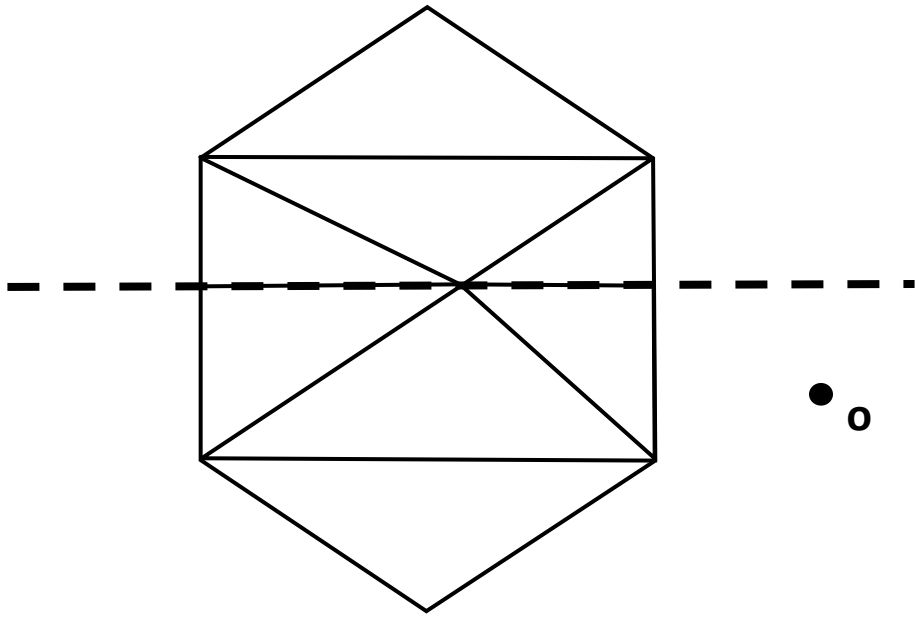
Submerged volume calculation
1st attempt



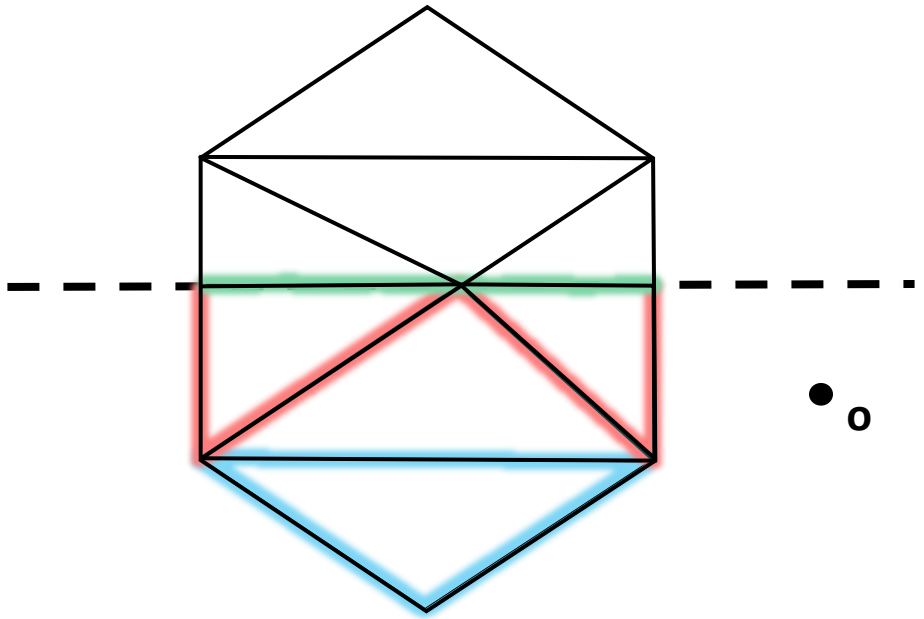
Submerged volume calculation
1st attempt



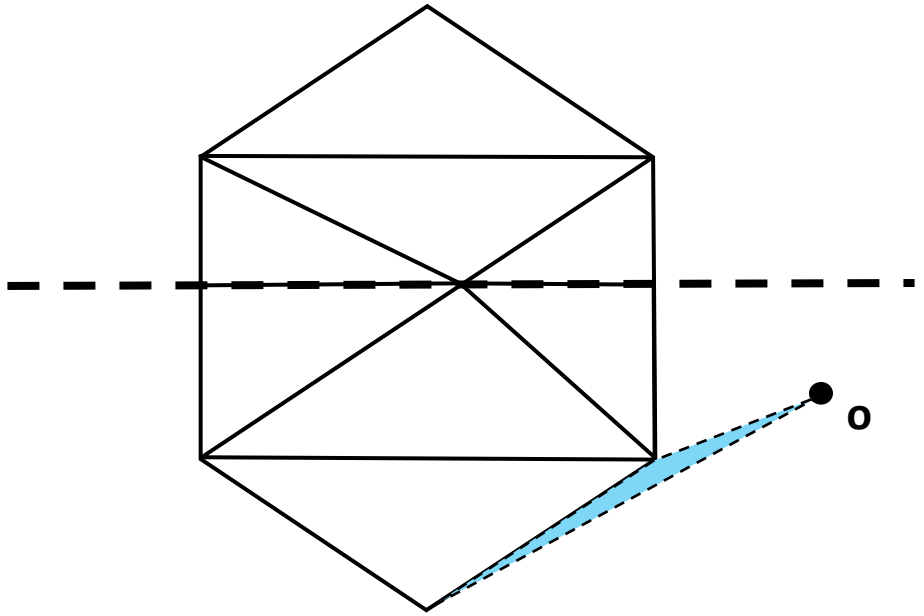
Submerged volume calculation
1st attempt



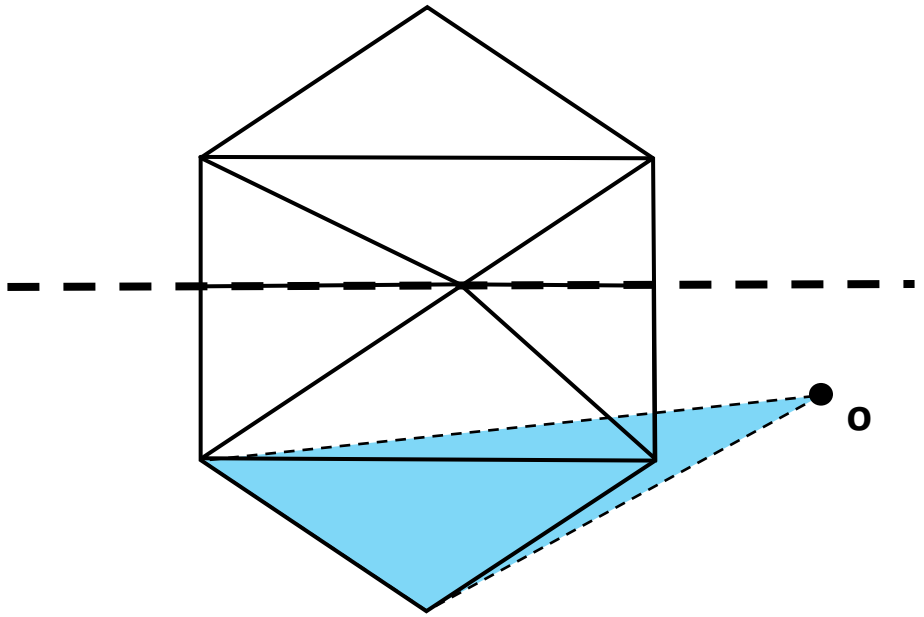
Submerged volume calculation
1st attempt



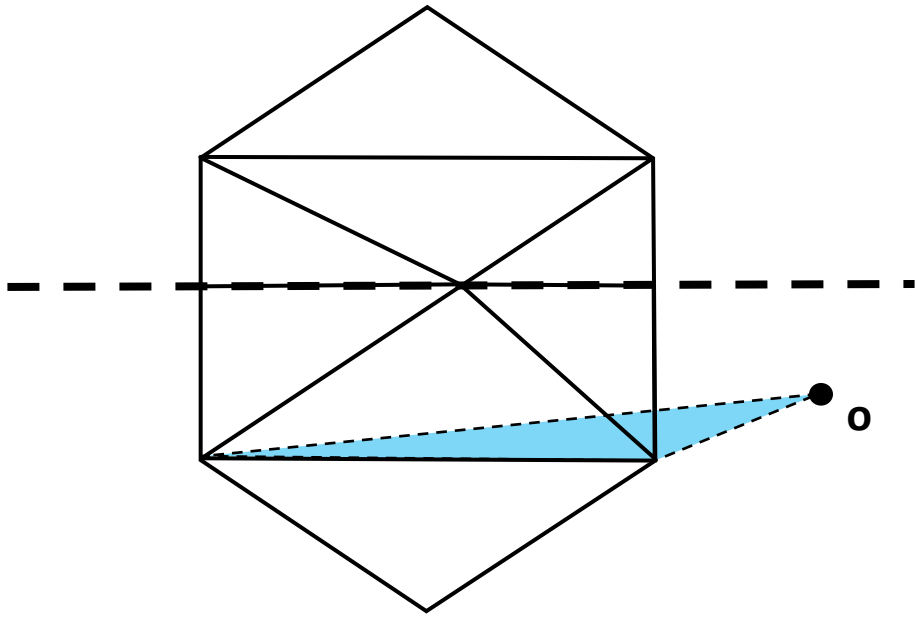
Submerged volume calculation
1st attempt



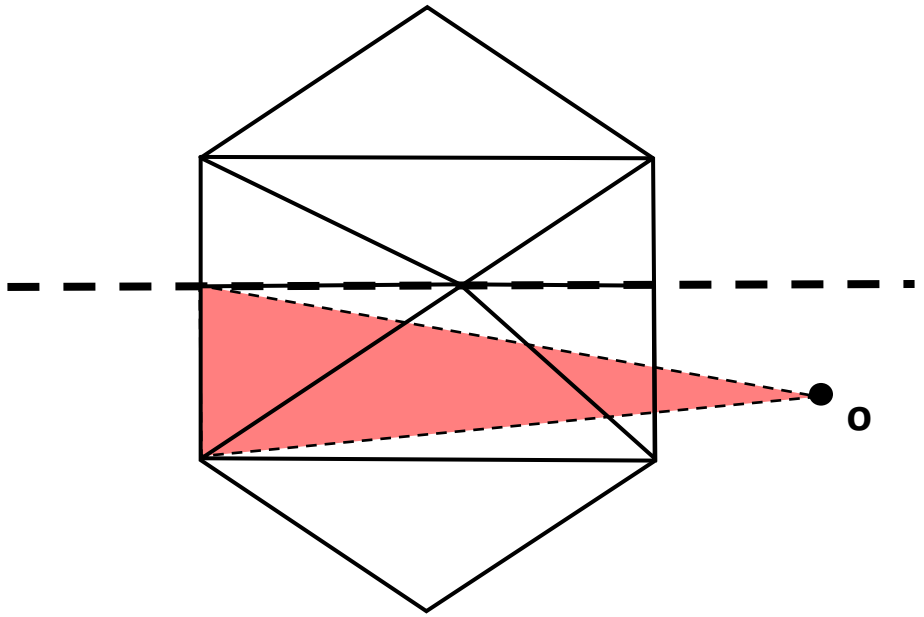
Submerged volume calculation
1st attempt



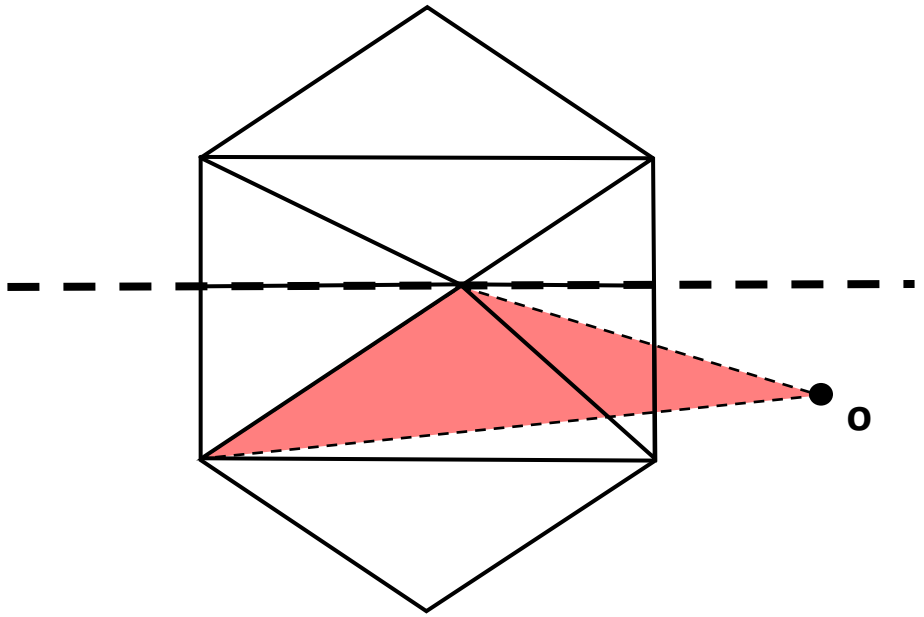
Submerged volume calculation
1st attempt



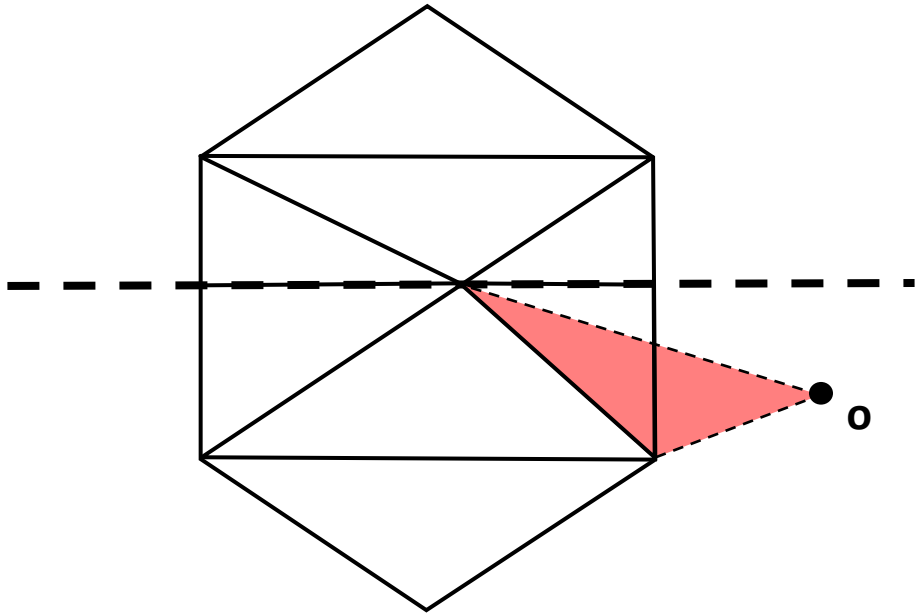
Submerged volume calculation
1st attempt



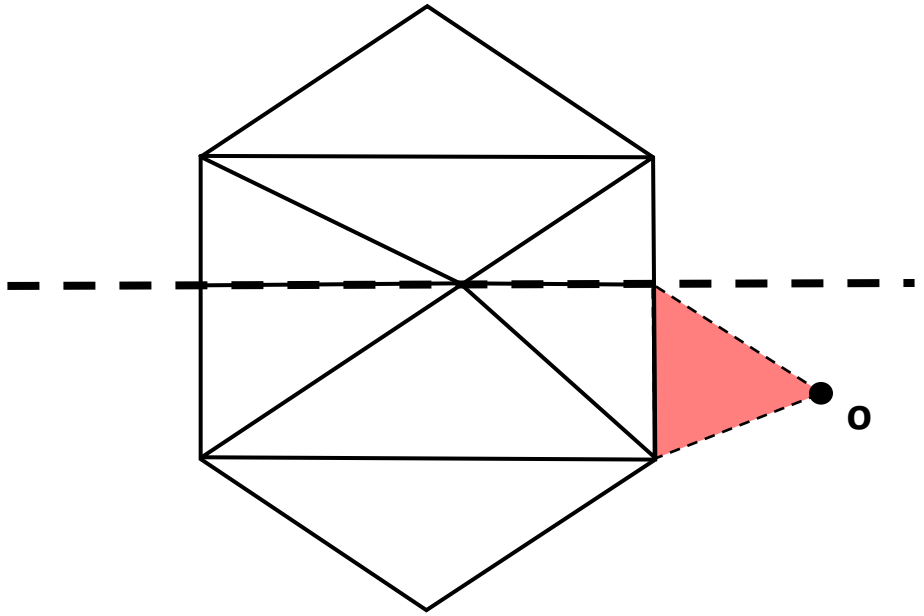
Submerged volume calculation
1st attempt



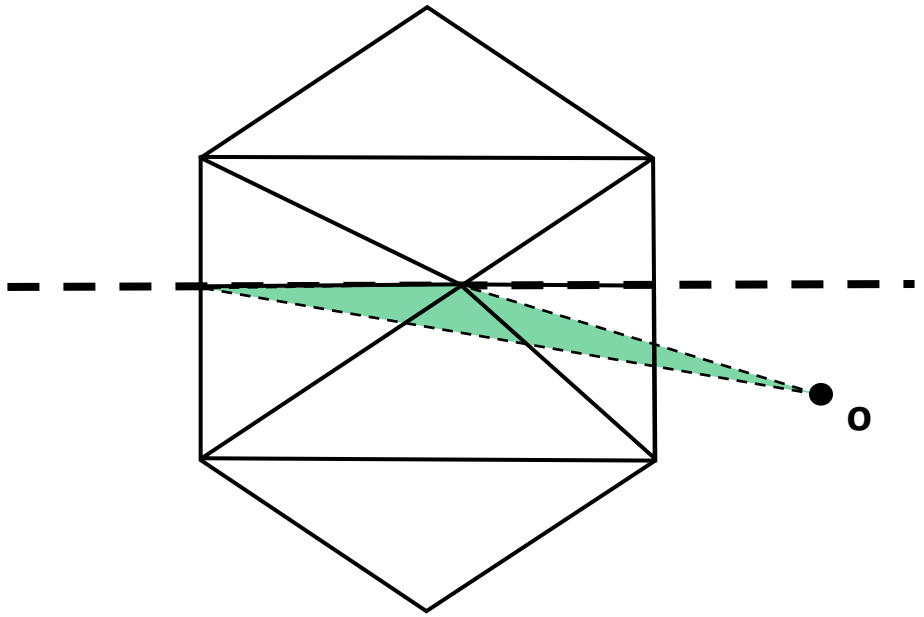
Submerged volume calculation
1st attempt



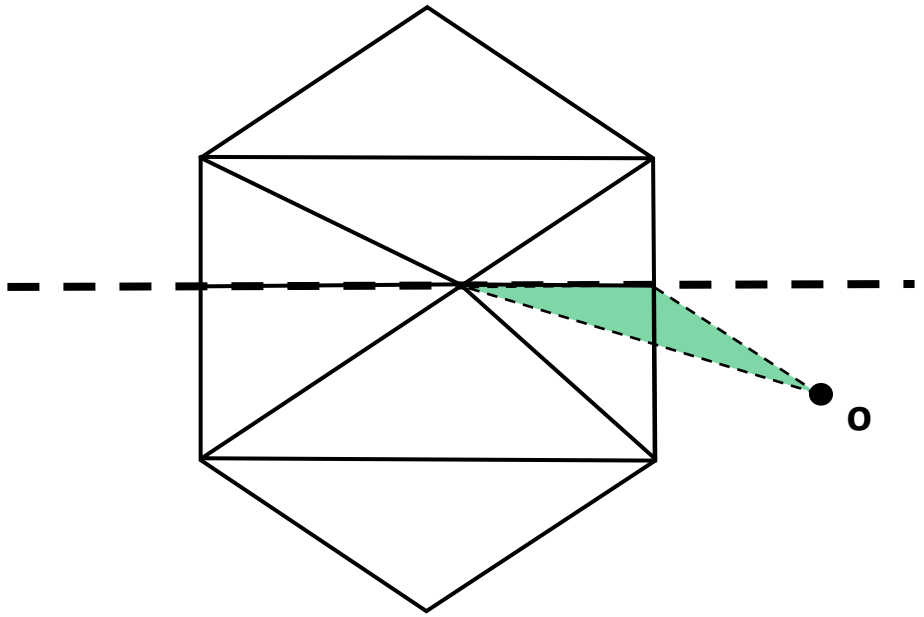
Submerged volume calculation
1st attempt



Submerged volume calculation
1st attempt



Submerged volume calculation
1st attempt



Volume and centroid of meshes

$$\mathrm{vol}(\Omega) = \iiint_{\Omega} 1 \, \mathrm{d}V = \frac{1}{3} \oiint_{\partial\Omega} \mathbf{id} \cdot \mathrm{d}\mathbf{S}$$

Integrate over a **closed** surface

Sum of triangle contributions

$$V = \sum_i V_i(\mathbf{o})$$

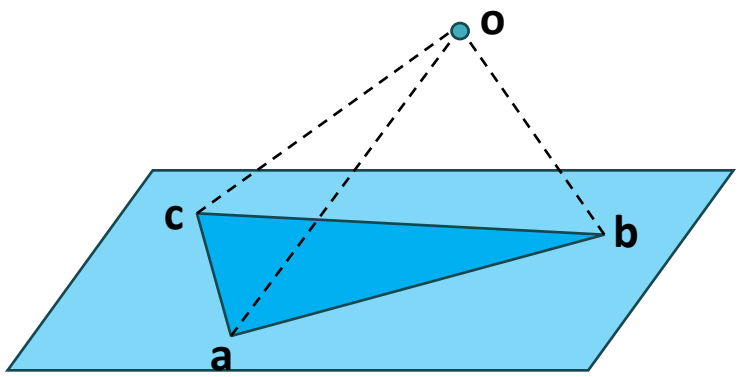
$$V_i(\mathbf{o}) := \frac{1}{6}(\mathbf{a}_i - \mathbf{o}) \cdot \Big((\mathbf{b}_i - \mathbf{o}) \times (\mathbf{c}_i - \mathbf{o}) \Big)$$

Volume element

$$\mathbf{C} = \frac{1}{V} \sum_i V_i(\mathbf{o}) \mathbf{C}_i(\mathbf{o})$$

$$\mathbf{C}_i(\mathbf{o}) = \frac{\mathbf{a}_i + \mathbf{b}_i + \mathbf{c}_i + \mathbf{o}}{4}$$

Centroid element



V≠0

Volume and centroid of meshes

$$\mathrm{vol}(\Omega) = \iiint_{\Omega} 1 \, \mathrm{d}V = \frac{1}{3} \oiint_{\partial\Omega} \mathbf{id} \cdot \mathrm{d}\mathbf{S}$$

Integrate over a **closed** surface → Sum of triangle contributions

$$V = \sum_i V_i(\mathbf{o})$$

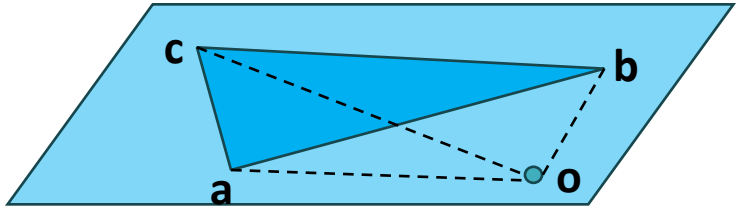
$$V_i(\mathbf{o}) := \frac{1}{6}(\mathbf{a}_i - \mathbf{o}) \cdot \Big((\mathbf{b}_i - \mathbf{o}) \times (\mathbf{c}_i - \mathbf{o}) \Big)$$

Volume element

$$\mathbf{C} = \frac{1}{V} \sum_i V_i(\mathbf{o}) \mathbf{C}_i(\mathbf{o})$$

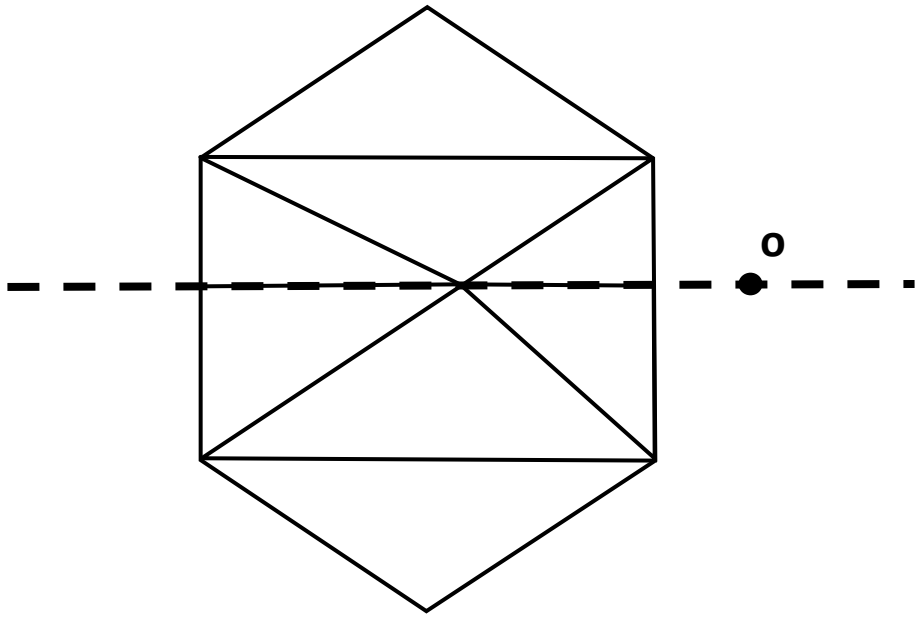
$$\mathbf{C}_i(\mathbf{o}) = \frac{\mathbf{a}_i + \mathbf{b}_i + \mathbf{c}_i + \mathbf{o}}{4}$$

Centroid element

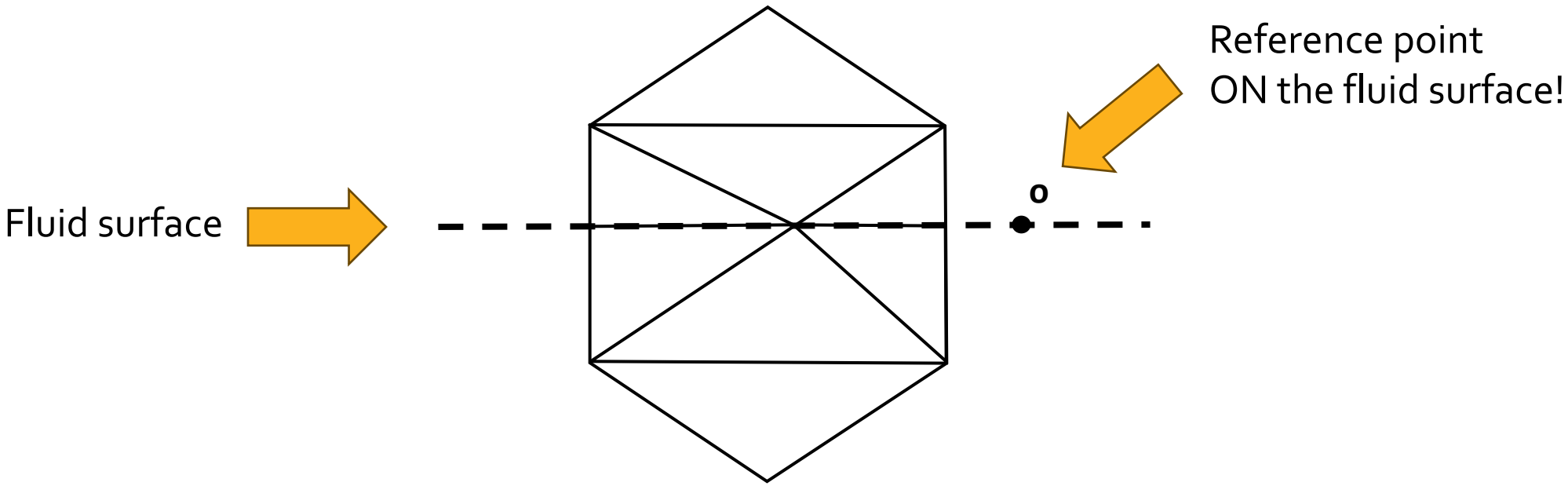


Zero volume! → **V=0**

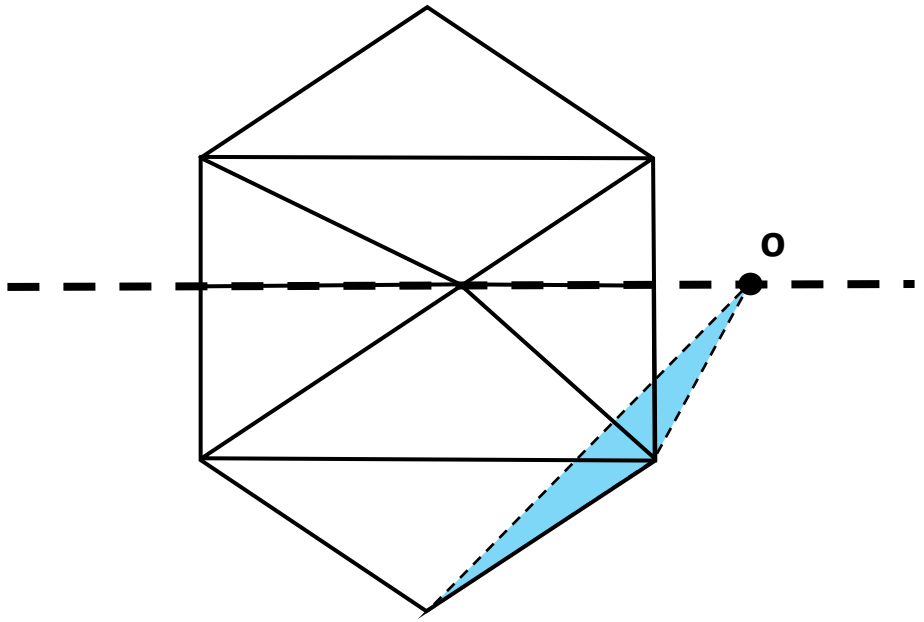
Submerged volume calculation 2nd attempt



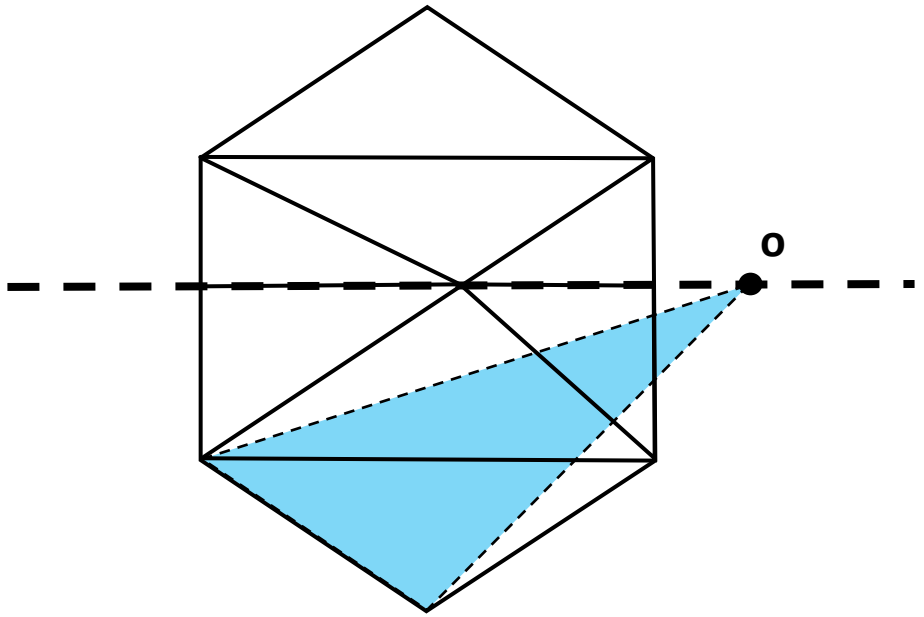
Submerged volume calculation 2nd attempt



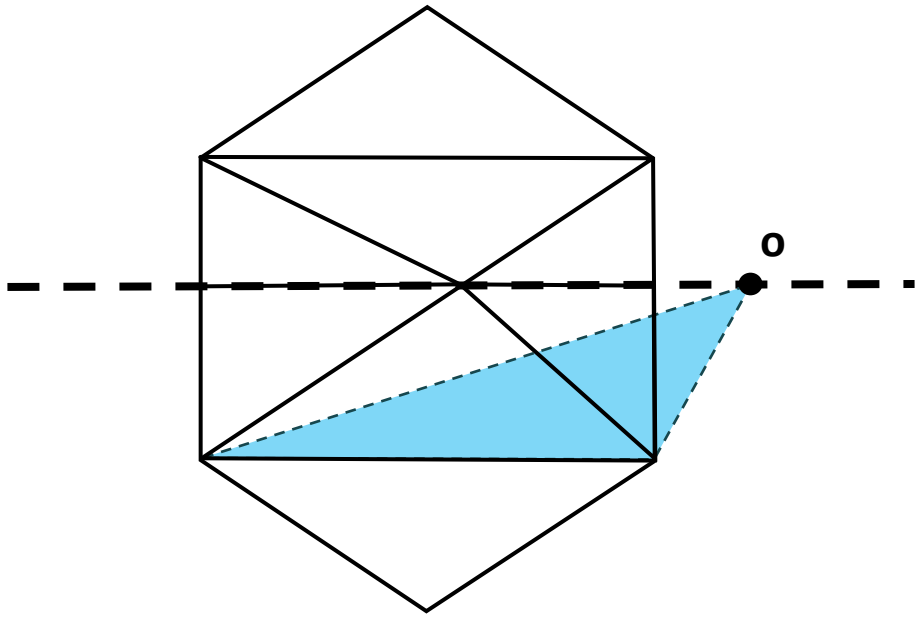
Submerged volume calculation 2nd attempt



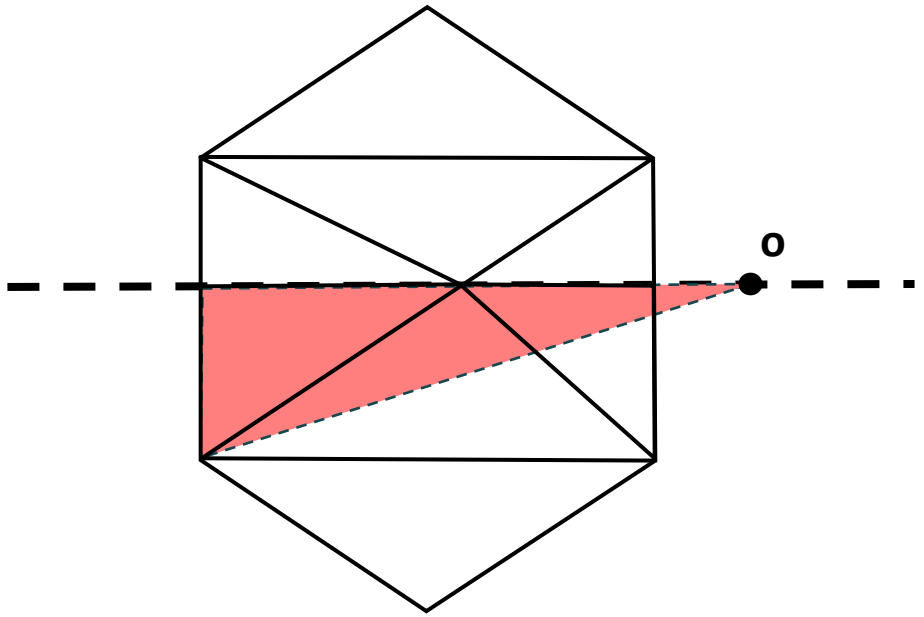
Submerged volume calculation 2nd attempt



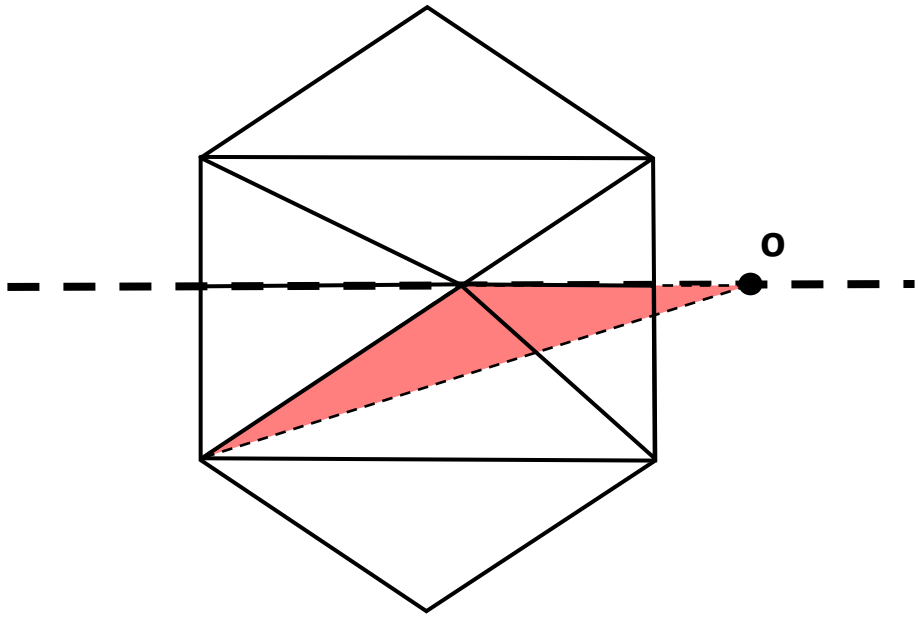
Submerged volume calculation 2nd attempt



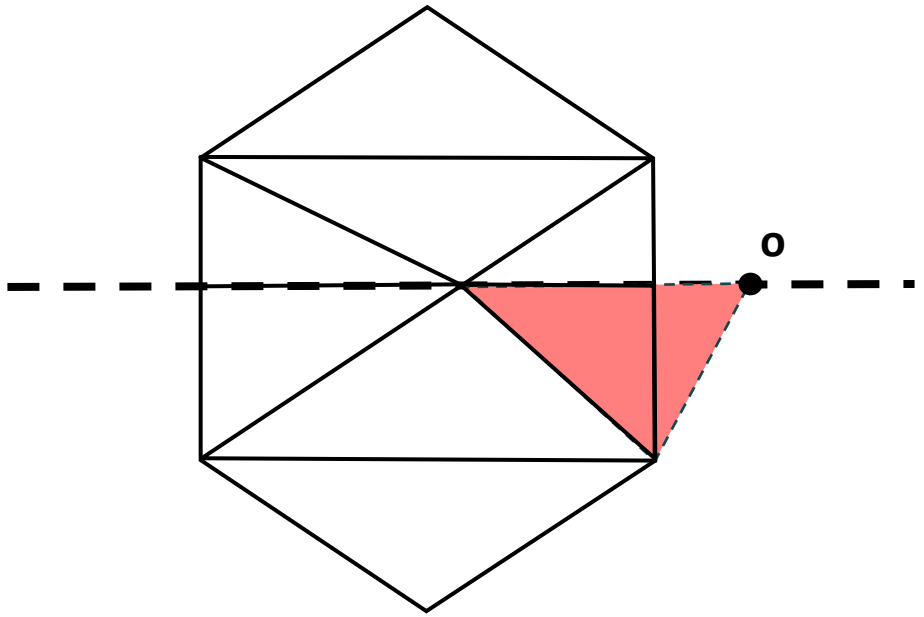
Submerged volume calculation 2nd attempt



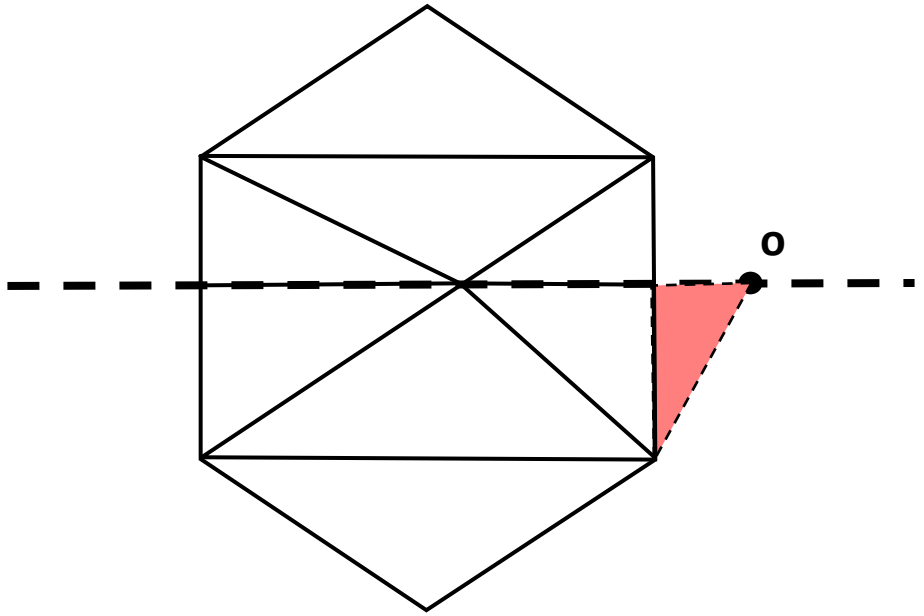
Submerged volume calculation 2nd attempt



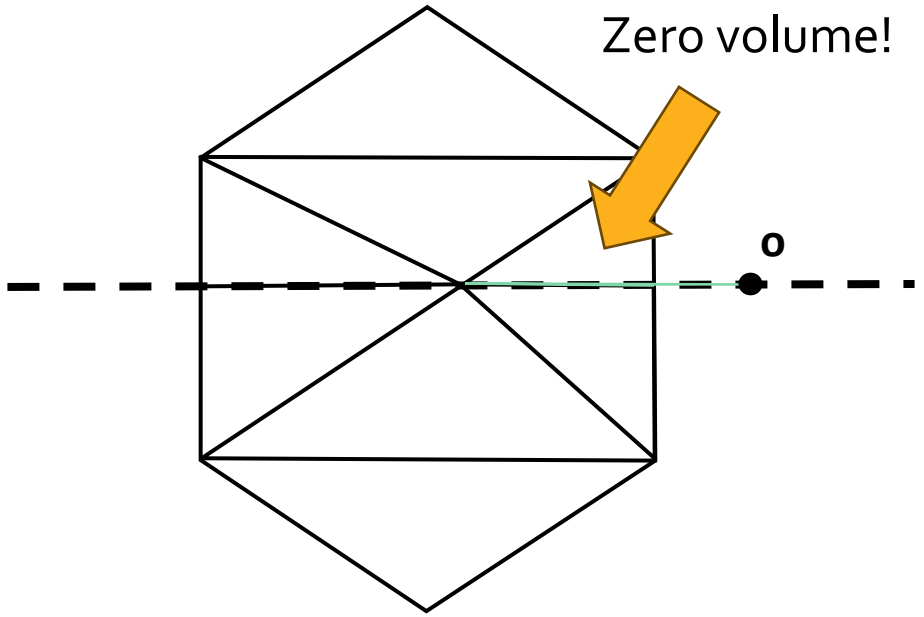
Submerged volume calculation 2nd attempt



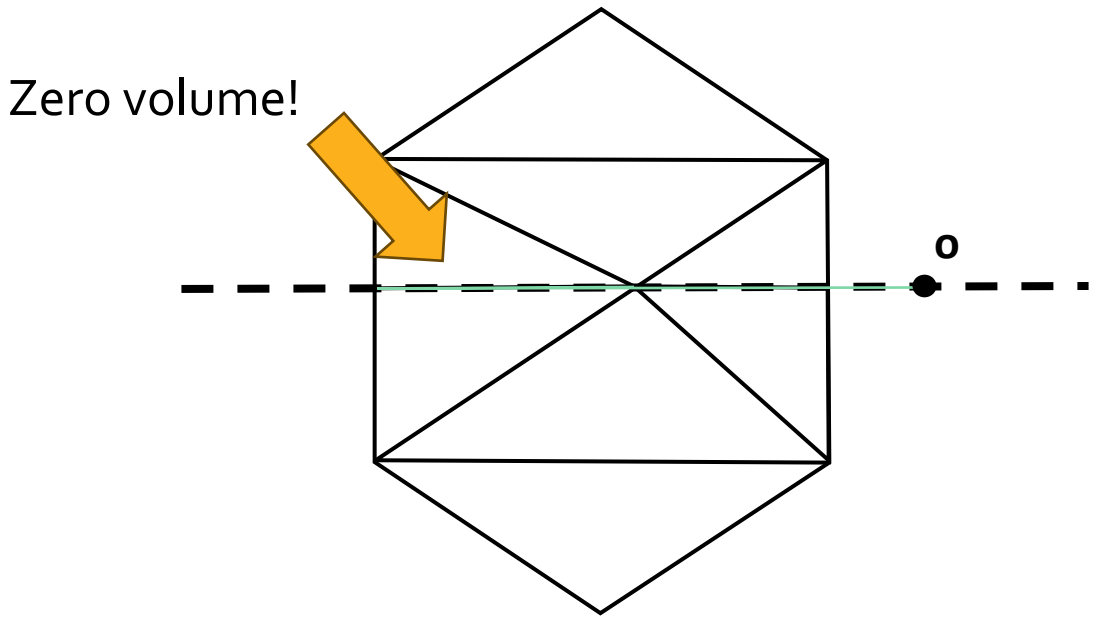
Submerged volume calculation 2nd attempt



Submerged volume calculation 2nd attempt



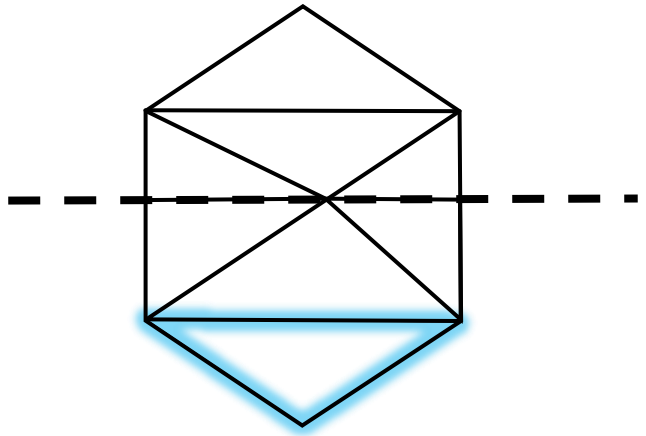
Submerged volume calculation 2nd attempt



Volume based algorithms

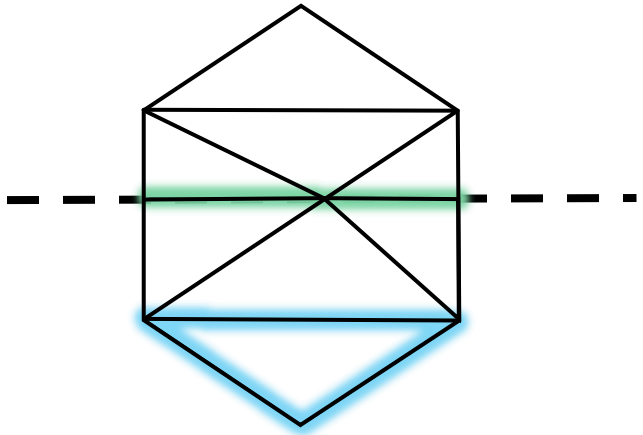
Fast

- Inaccurate on coarse meshes ☹
- Acceptable on fine meshes 😐
- Fast (precalculation) ☺
- Easy to implement ☺



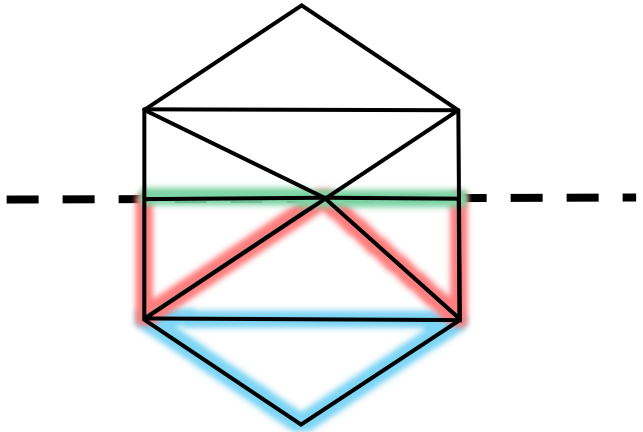
Improved

- Better on coarse meshes 😐
- Almost exact on fine meshes ☺
- Slower 😐
- Easy to implement ☺



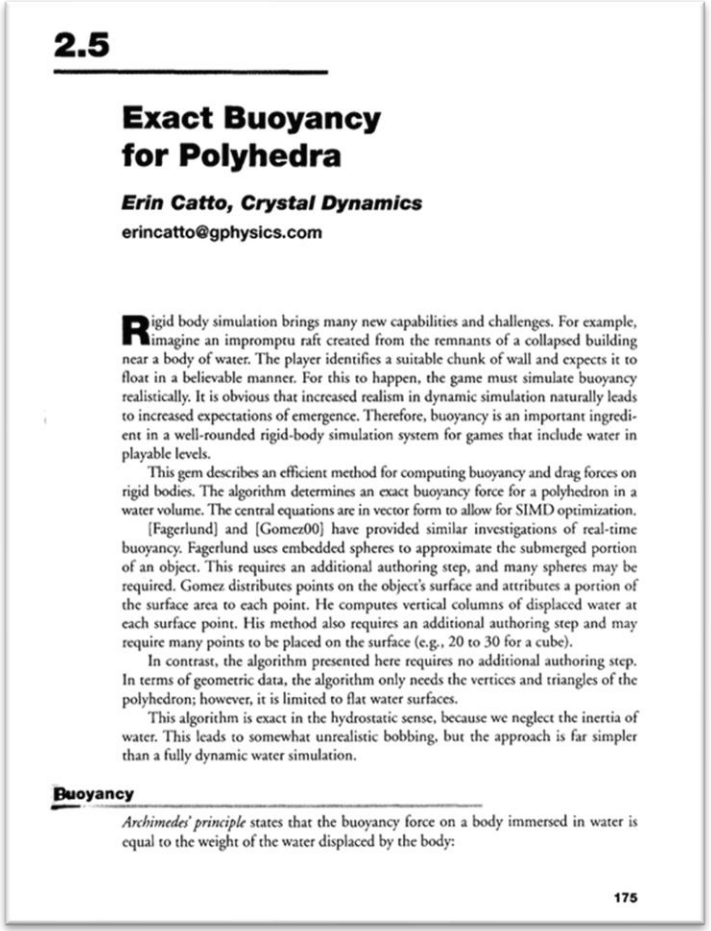
Exact

- Exact on coarse meshes ☺
- Exact on fine meshes ☺
- Computationally expensive ☹
- Hard to implement (edge cases) ☹



Moment of honesty

- I believed, the integration trick is my own invention.
- Under the revision process, it turned out, this is not true.
- Erin Catto: Exact Buoyancy for Polyhedra (GPG6, 2006)
- Exact algorithm, no approximations



2.5

Exact Buoyancy for Polyhedra

Erin Catto, Crystal Dynamics
erincatto@gphysics.com

Rigid body simulation brings many new capabilities and challenges. For example, imagine an impromptu raft created from the remnants of a collapsed building near a body of water. The player identifies a suitable chunk of wall and expects it to float in a believable manner. For this to happen, the game must simulate buoyancy realistically. It is obvious that increased realism in dynamic simulation naturally leads to increased expectations of emergence. Therefore, buoyancy is an important ingredient in a well-rounded rigid-body simulation system for games that include water in playable levels.

This gem describes an efficient method for computing buoyancy and drag forces on rigid bodies. The algorithm determines an exact buoyancy force for a polyhedron in a water volume. The central equations are in vector form to allow for SIMD optimization.

[Fagerlund] and [Gomez00] have provided similar investigations of real-time buoyancy. Fagerlund uses embedded spheres to approximate the submerged portion of an object. This requires an additional authoring step, and many spheres may be required. Gomez distributes points on the object's surface and attributes a portion of the surface area to each point. He computes vertical columns of displaced water at each surface point. His method also requires an additional authoring step and may require many points to be placed on the surface (e.g., 20 to 30 for a cube).

In contrast, the algorithm presented here requires no additional authoring step. In terms of geometric data, the algorithm only needs the vertices and triangles of the polyhedron; however, it is limited to flat water surfaces.

This algorithm is exact in the hydrostatic sense, because we neglect the inertia of water. This leads to somewhat unrealistic bobbing, but the approach is far simpler than a fully dynamic water simulation.

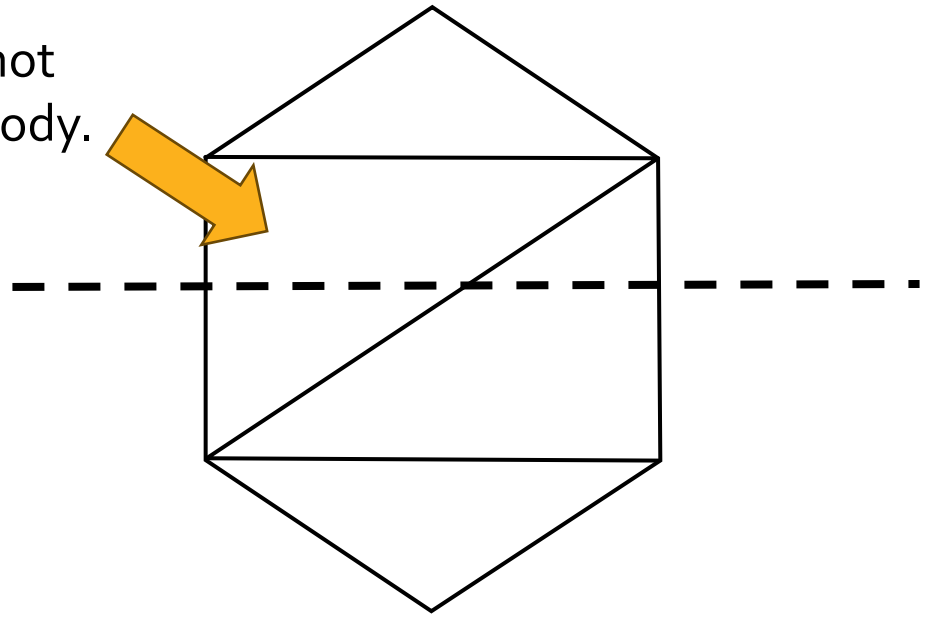
Buoyancy

Archimedes' principle states that the buoyancy force on a body immersed in water is equal to the weight of the water displaced by the body:

175

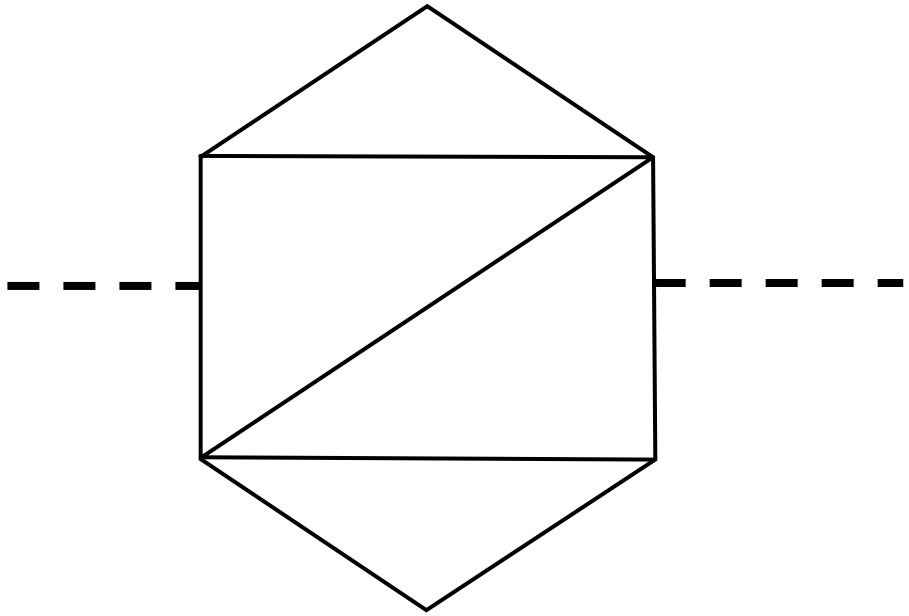
The physical problem

The fluid surface is not present **inside** the body.



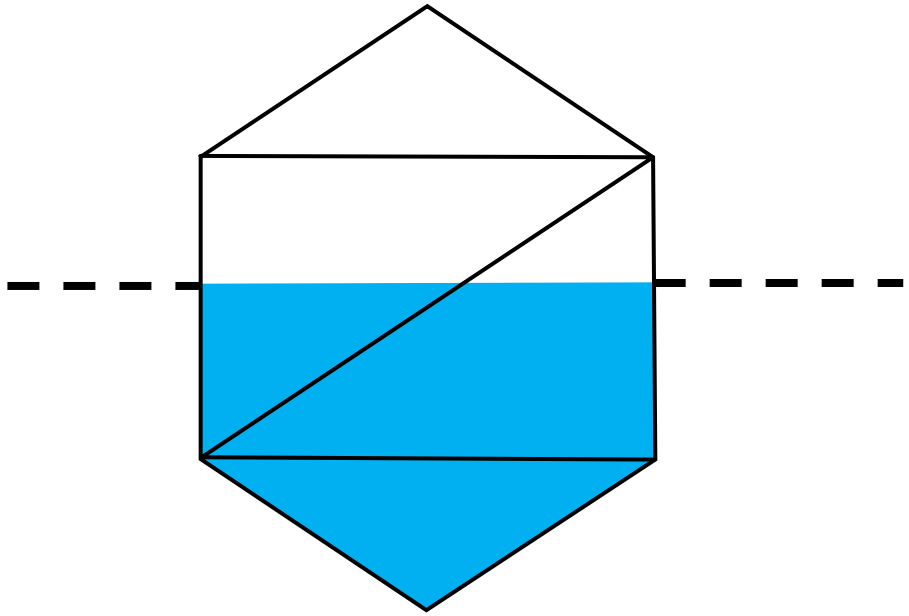
The physical problem

What is the submerged volume?



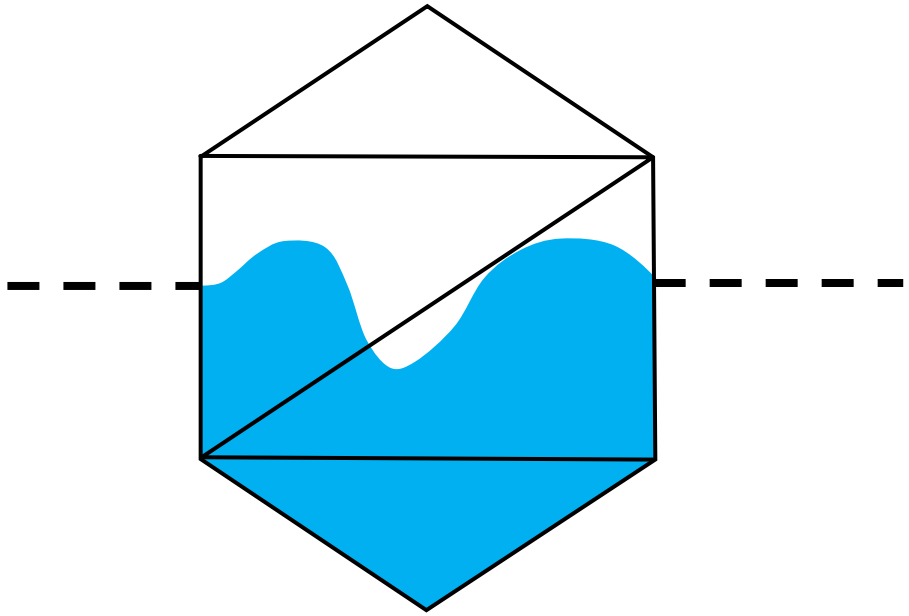
The physical problem

What is the submerged volume?



The physical problem

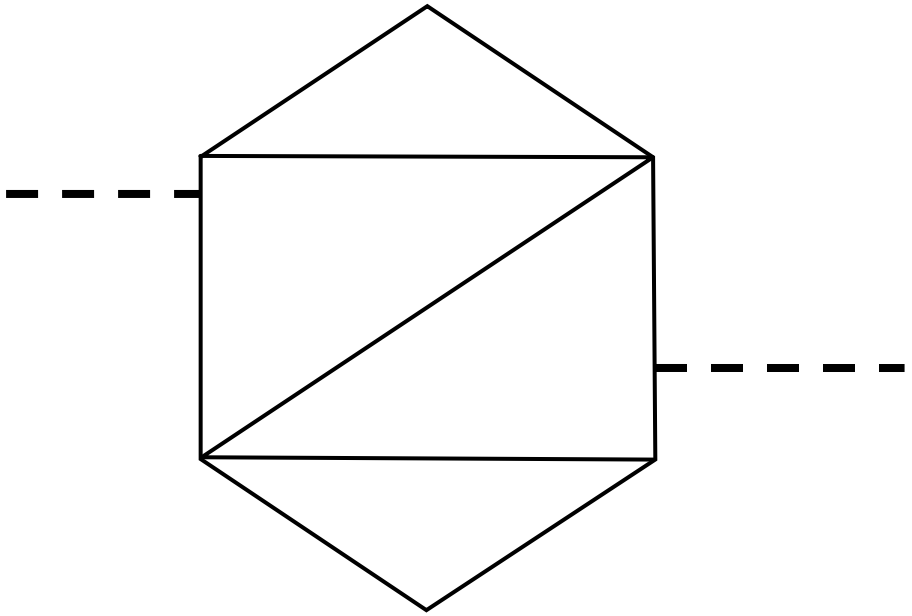
What is the submerged volume?



The physical problem

What is the submerged volume?

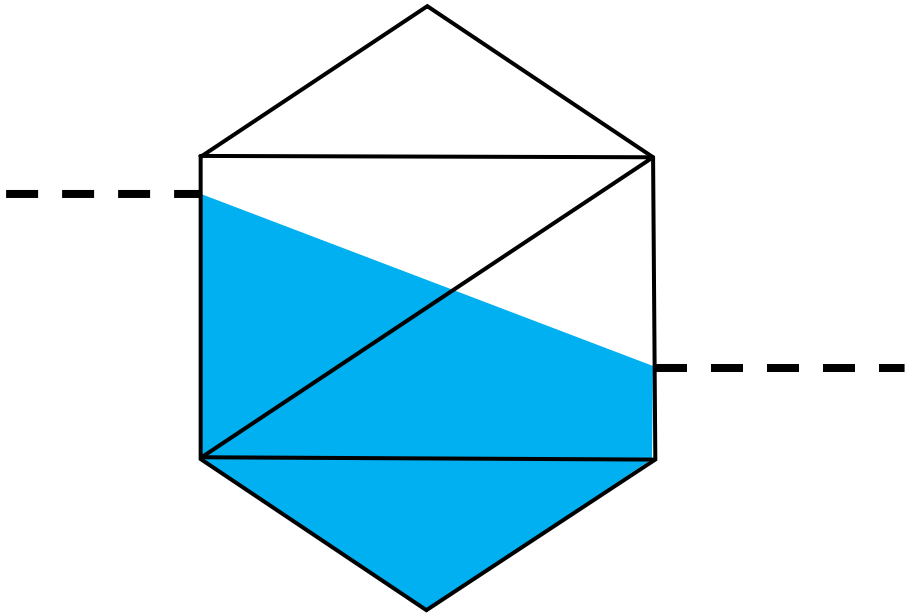
Waving surface?



The physical problem

What is the submerged volume?

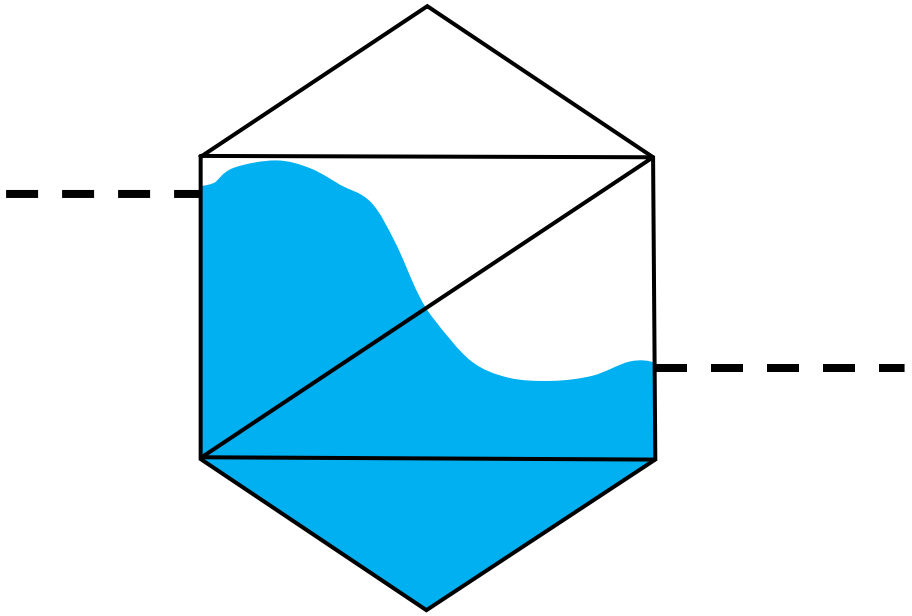
Waving surface?



The physical problem

What is the submerged volume?

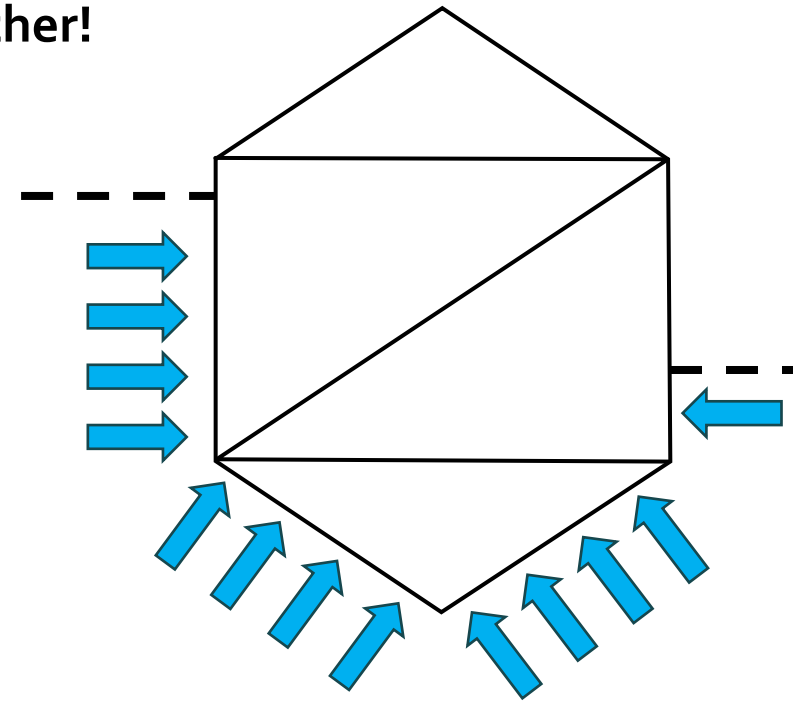
Waving surface?



The physical problem

We don’t know what is inside!

The fluid doesn’t know either!



$$\mathbf{F}_B = -\iint_\Sigma p \, d\mathbf{S} = \iint_\Sigma \rho g y \, d\mathbf{S}$$

$$\mathbf{F}_i = \rho g (\mathbf{C}_i)_y \mathbf{A}_i$$

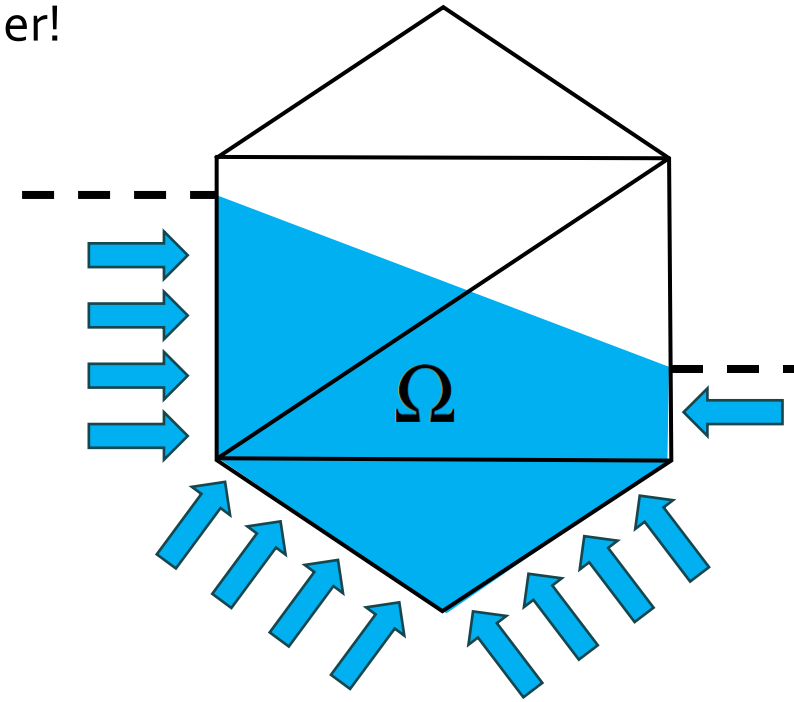
Force elements

Integrate over a **non-closed** surface

Physical problem

We do not know what is inside!

The fluid doesn't know either!



$$\mathbf{F}_B = -\iint_\Sigma p \, d\mathbf{S} = \iint_\Sigma \rho g y \, d\mathbf{S}$$

$$\mathbf{F}_i = \rho g (\mathbf{C}_i)_y \mathbf{A}_i$$

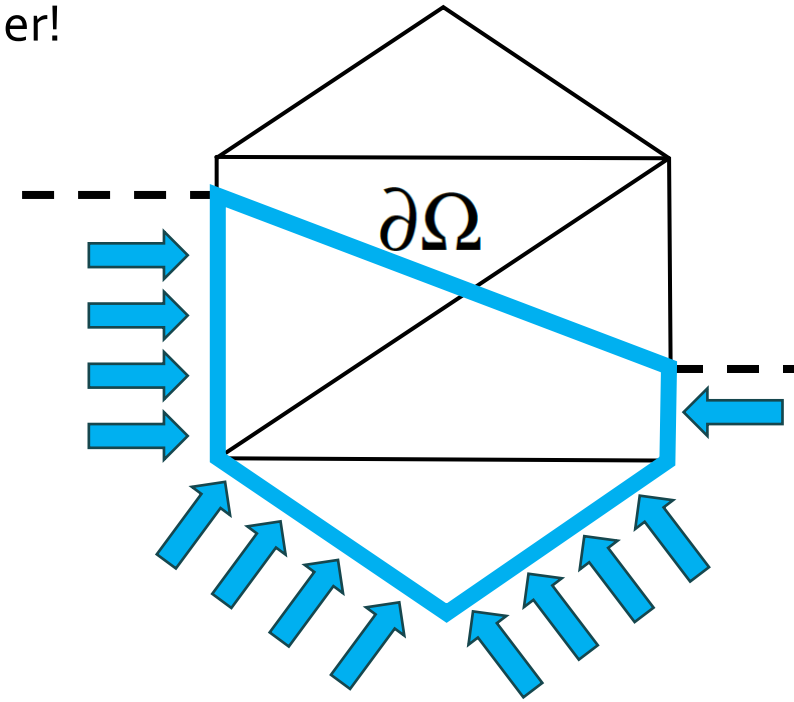
Force elements

Integrate over a **non-closed** surface

Physical problem

We do not know what is inside!

The fluid doesn't know either!



$$\mathbf{F}_B = -\iint_\Sigma p \, \mathrm{d}\mathbf{S} = \iint_\Sigma \rho g y \, \mathrm{d}\mathbf{S}$$

$$\mathbf{F}_i = \rho g (\mathbf{C}_i)_y \mathbf{A}_i$$

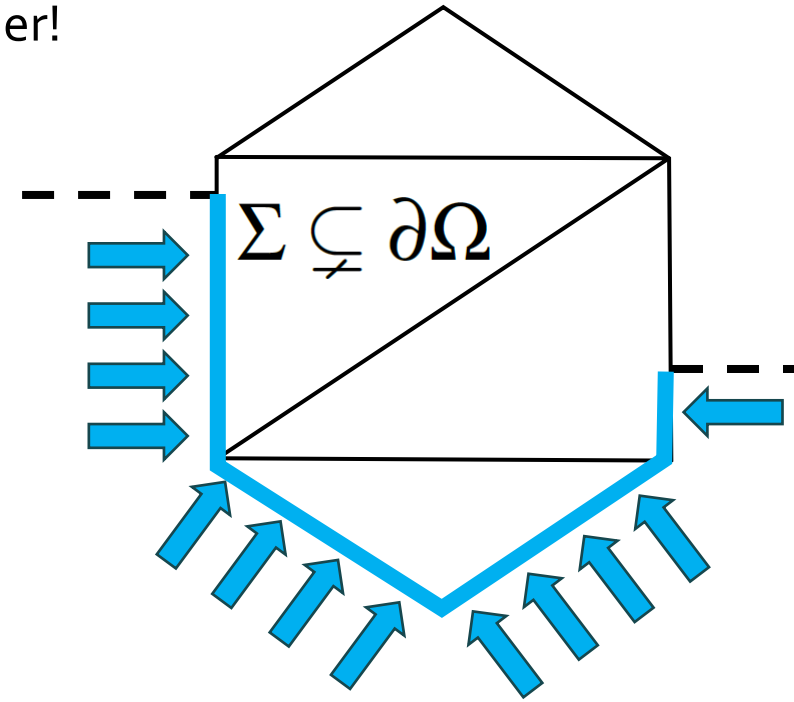
Force elements

Integrate over a **non-closed** surface

Physical problem

We do not know what is inside!

The fluid doesn't know either!



$$\mathbf{F}_B = -\iint_\Sigma p \, d\mathbf{S} = \iint_\Sigma \rho g y \, d\mathbf{S}$$

$$\mathbf{F}_i = \rho g (\mathbf{C}_i)_y \mathbf{A}_i$$

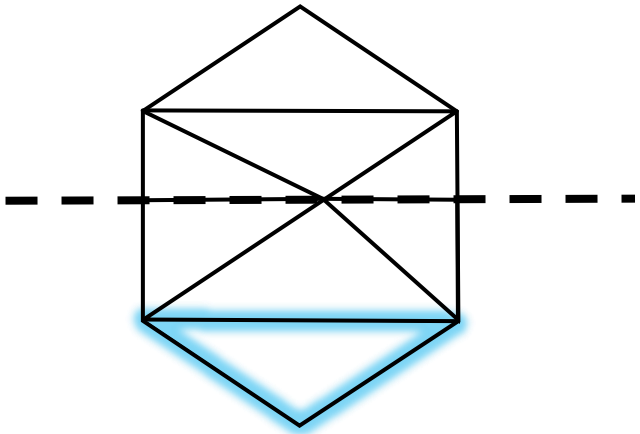
Force elements

Integrate over a **non-closed** surface

Surface based algorithms

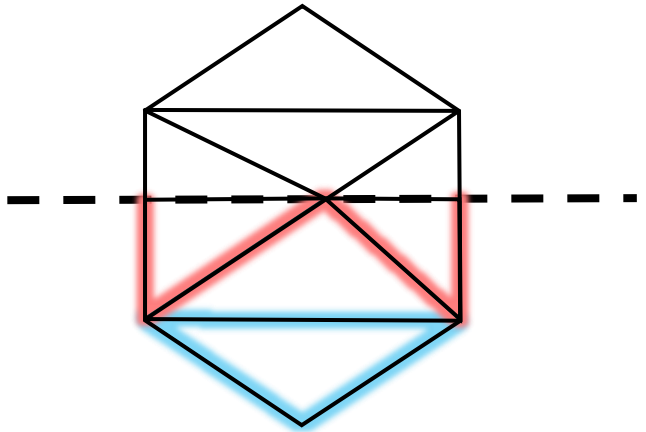
Fast Surface

- Inaccurate on coarse meshes ☹
- Acceptable on fine meshes 😐
- Not so fast (precalculation) 😐
- Easy to implement ☺

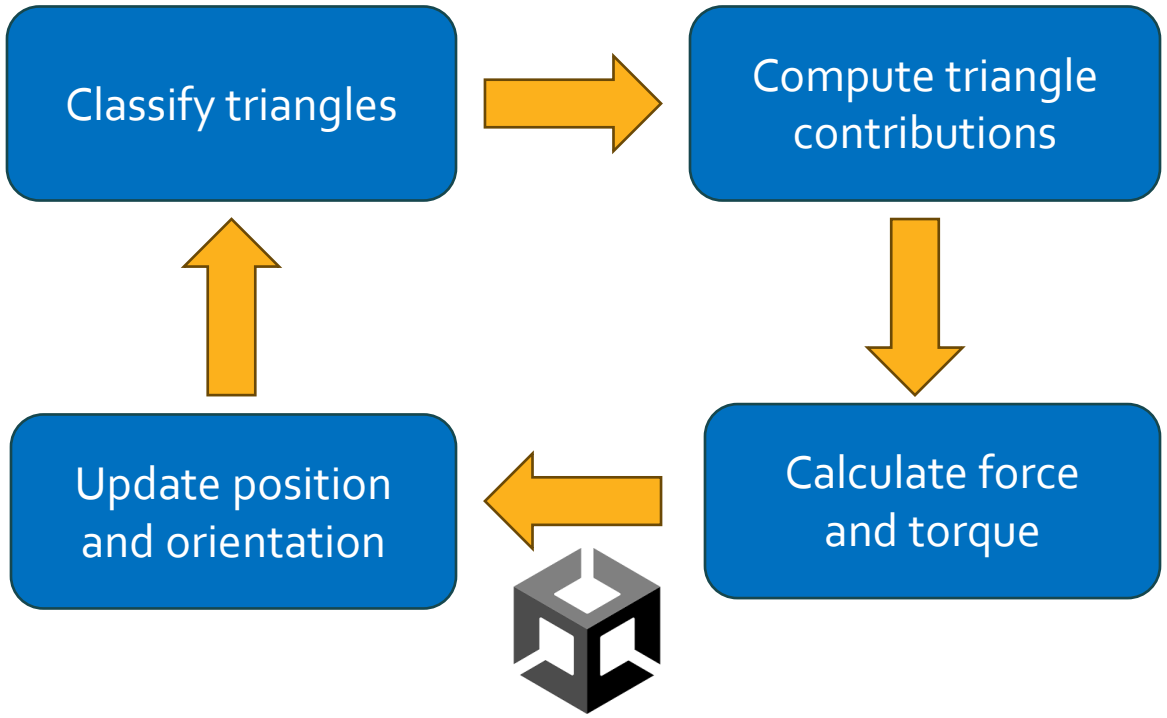


Exact Surface

- Exact on coarse meshes ☺
- Exact on fine meshes ☺
- Computationally expensive ☹
- Hard to implement (edge cases) ☹



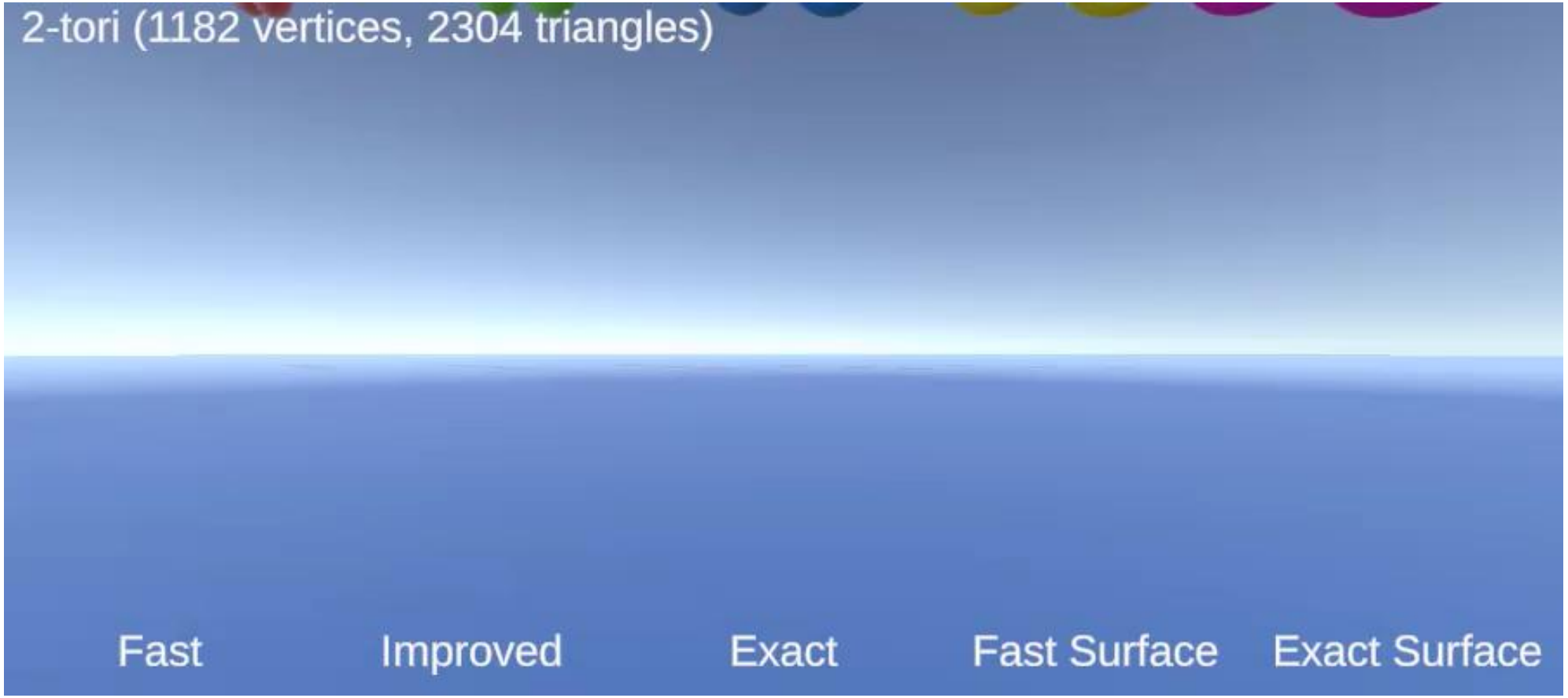
Simulation workflow



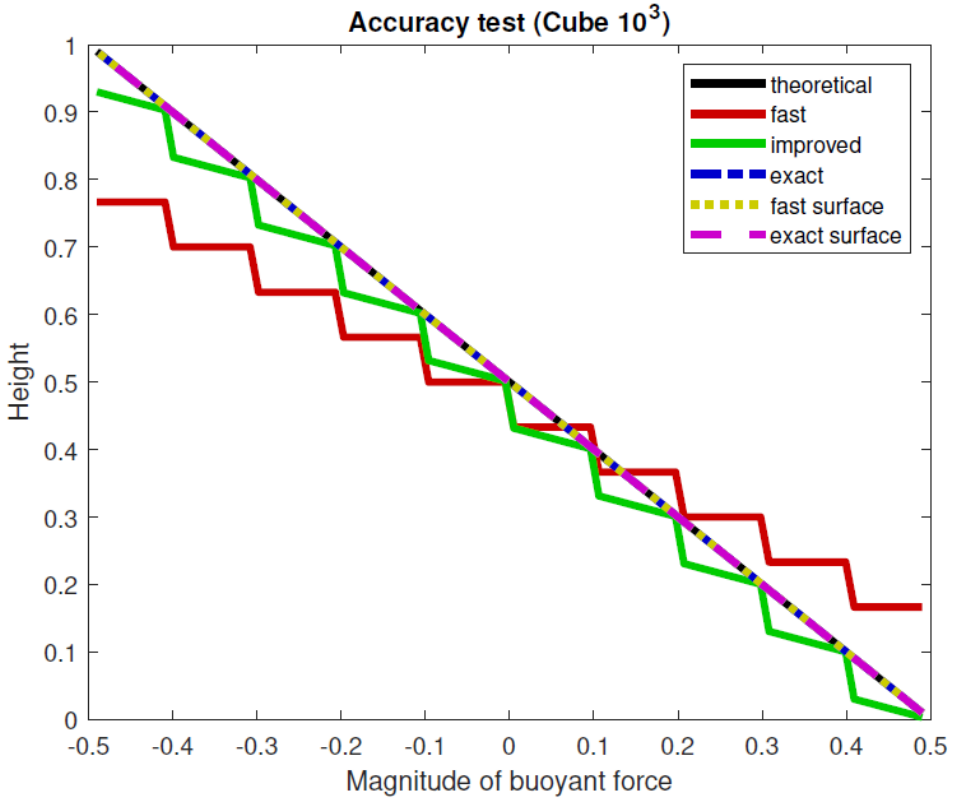
Simulation

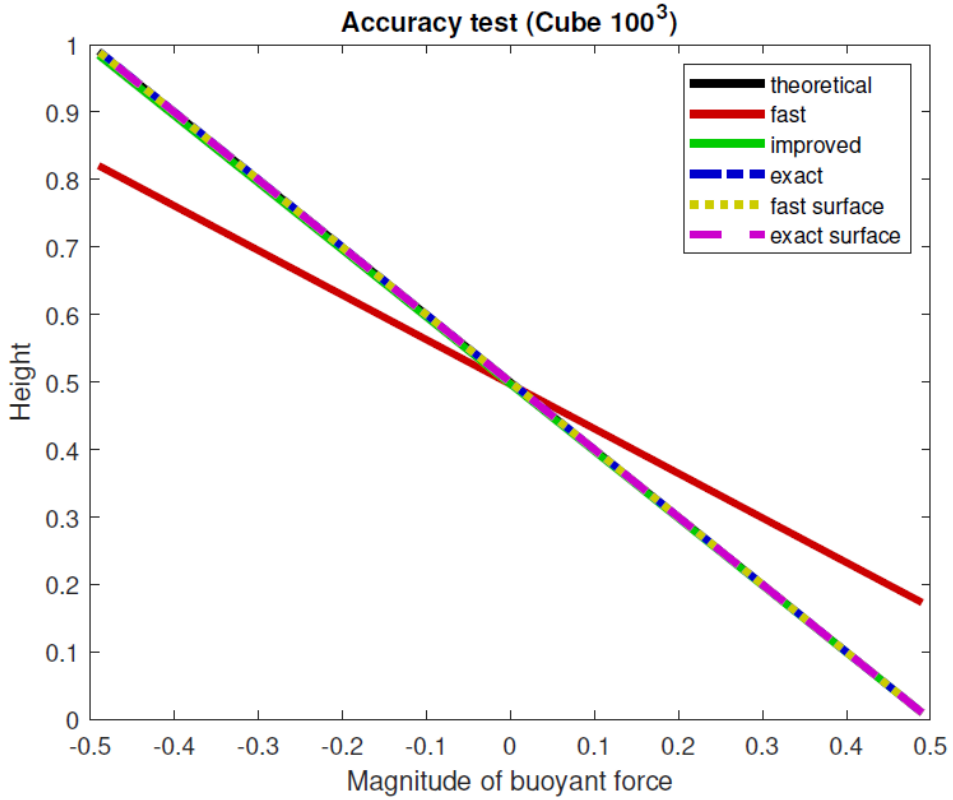
Torusknot (880 vertices, 1760 triangles)
Fast
Improved
Exact
Fast Surface
Exact Surface

Simulation


2-tori (1182 vertices, 2304 triangles)
Fast
Improved
Exact
Fast Surface
Exact Surface

Accuracy tests





Performance tests

Total time cost for buoyancy calculation in 1 physics step

	Torusknot [ms]	Bunny [ms]	Armadillo [ms]
Fast	0.09	0.23	0.98
Improved	0.18	0.69	2.96
Exact	0.26	0.75	3.05
Fast Surface	0.22	0.76	3.45
Exact Surface	0.33	0.90	3.94

Limitations & future plans

Mesh
- Rigid body
- Uniform density

Fluid
- Incompressible
- Planar surface

Future
- Compute drag forces in the fluid
- Generalization for waving surface
- Body's effect on the fluid