Simulating Buoyancy of Different Spheres in a Given Fluid and their Roles in Self-Organization: Is Neutral Buoyancy Possible?

Henrique Matos (64369) [1,2] and Duarte Gonçalves (64465)[1,2]

[1]Faculty of Sciences of the University of Lisbon, Cidade Universitária
[2]Artificial Life, M.E.I.

Abstract

In this paper we observe and analyze how different balls of different weights and sizes, and consequently different densities, are affected by the density of the liquid they're being thrown into, correlating the effects of buoyancy[3] with drag force [11] and providing a user-interactive simulation for mathematical educative purposes, being the user able to check how the different balls achieve a state of equilibrium where they do not sink neither emerge within the liquid, providing a brief overlook within Self-Organization in Artificial Life subjects. It is explained how buoyancy works as a characteristic of submerged objects, be them fully or partially, and how density plays a big role in it, considering drag force as the main reaction to a body that is being moved inside a fluid through the action of gravity (the action part of the action-reaction pair), taking into account the necessity to move fluid particles out of the way, being the latter responsible for the resistance within the fluid. We take into account the simulation results and consequently deduce which weight/radius values and water density are needed to achieve neutral buoyancy. All of this done through an application developed by the investigative group [8].

Introduction

buoyancy has first been described through the Archimedes Principle in a way that the weight of the body thrown within water would displace a certain amount of liquid regarding its own materials. Therefore,if a body made of a certain material X would be heavier than the material Y, being both of the same size, the body made out of material X would displace more water than material Y.

In modern physics it is also taken into consideration that buoyancy must relate gravity and, consequently, drag force, since an object falling within a liquid isn't simply acting according to material density but interacting with the former according to its very own density and gravitational force. A body within the liquid is also being subjected to the a certain drag force, that acts as a reaction force to the gravitational attraction responsible for the falling motion (hence the action-reaction pair).

It has been created a graphical simulation that acts upon such principles of buoyancy, enabling a user to interact with the system in a way that would change liquid density and object size, weight and radius (and also color for easier distinction). In order to create such a system, the Java programming language has been used alongside the Processing Framework [9], providing the latter various tools for graphical interfaces and physics based applications development. With these tools available, both back-end and front-end has been coded in a way that would allow for each to be modular yet independent, providing a GUI crucial for observing the simulation. In a way that would enable users to change simulation parameters in real-time, it has been used the ControlP5 library from Processing [9], enabling for GUI components like sliders, text fields and buttons to be added upon the simulation, adding and fetching values that would be introduced as parameters in real time. The user needs only to play with the sliders, sit back and observe the simulation of an object's buoyancy trying to reach a state of equilibrium via Self-Organization [7], which means, until it reaches a fixed point where it neither sinks to the bottom nor emerges to the surface, being able to check how it would float up or down until reaching **neutral buoyancy**.

Buoyancy and its Principles

Buoyancy is known as the tendency of an object to float or to rise in a fluid when submerged, be it liquid or gas [3]. Taking the famous Greek mathematician and physicist, Archimedes, into account, the weight of the water that the object displaces when it is placed in water must be greater than the weight of the object itself. The "Principle of Archimedes" has been the basis of buoyancy for a long time, however, such concept has never taken into account gravitational forces, since it was yet to be conceptualized during its time of conception.

In order to explain buoyancy, one must resort to *gravity*. Acting upon all objects, when placed in a fluid, the latter must supply a force equal in magnitude but opposite in direction to the gravitational force for the objects to float, therefore, it's deduced the **Buoyant Force**. This force is actually closely tied to the *density* of a fluid, having the formula [1]:

1. ρ - Density
2. m - Mass of object
3. V - Volume of object

$$\rho = \frac{m}{V} \ , \tag{1}$$

Relating an object's density to the density of the fluid it is within, is called *specific gravity* [2] (defined as "ratio of the density of a substance to that of a standard substance). When an object has lower specific gravity than the fluid in question, it floats. It is actually important to mention that objects with great buoyancy are objects with **large volumes** and a **relatively low density**[3].

Necessary Calculations for Buoyant Force

Knowing buoyancy as being defined at the upward force exerted on an object [3], as a result of such object being totally (or even partially) submerged in a fluid, this force is equal to the weight of the fluid displaced by the object. In order to calculate this force it is needed volume of displaced fluid (cubic meters), density (kilograms per cubic meter) and gravity [10].

1. F_b - Buoyant force
2. ρ_f - Density
3. V_d - Volme of displaced liquid
4. g - Gravitational force

$$F_b = \rho_f V_d g \ , \tag{2}$$

Buoyancy itself is the characteristic responsible for the floating characteristic of the objects. However, on its own it cannot define the behavior of the submerged object. It relies on **drag**, the vertical upward force due to fluid weight, being this the resistant force, and **lift**, the force perpendicular to the direction of motion.

Drag force is responsible for the slowing down of an object in the submerged fluid [4], due to the displacing of particles within the former, given that such particles exert a force upon the object (Newton's Third Law - "To every action, there is always opposed an equal reaction"). In order to be calculated, it is needed:

1. C_D - Drag coefficient
2. ρ - Density of the fluid
3. V^2 - Velocity of the object relative to the fluid
4. A - Area of the object

$$F_D = \frac{1}{2} C_D \rho v^2 A \ , \tag{3}$$

In this paper, for *simulation and simplification purposes*, the drag coefficient has been defined with a value of 1. In the real world, this would mean that with respect to hydrodynamics efficiency, the object would have a high drag, displaying poor hydrodynamics [5].

Case Study: Dropping Spheres in a Fluid and related Self-Organization

Taking all the above into reference, it is wanted to determine how *spheres of different weights* behave when falling into a fluid.

Having spheres of varying weights drop into the fluid would better understand the parameters needed to achieve a state of equilibrium, where the sphere neither sinks to the bottom nor floats to the top, but stays floating in a fixed point within the middle of the fluid. Having defined, preemptively, the weights of the spheres and the density of the liquid, it is wanted to observe how such spheres would achieve balance within the system.

Therefore, this experiment falls into the realm of **Self-Organization**, defined as "spontaneous ordering tendencies in complex systems" [6]. In this case study, no external control is added, order is increased (until the spheres achieve a final static, or almost static, velocity), adaptability is present due to the very core principles of buoyancy (and, consequently, drag force), spheres interact with the fluid and is asynchronous (not globally coordinated, the fluid is present, and the forces themselves are responsible for the desired stability) [7].

For this case study, the varying scenarios of a sphere being dropped into the fluid must be taken into account:

1. Sphere has less density than the fluid itself
   - Theoretical result - the sphere floats within the surface of the fluid, bouncing before stabilizing
2. Sphere is more dense than the fluid itself
   - Theoretical result - the sphere immediately sinks into the bottom of the fluid
3. Sphere has the same density as the fluid
   - Theoretical results - it achieves stability, also known as **neutral buoyancy**, neither sinking nor floating but staying on the same position within the liquid

It is wanted, through simulation, to achieve the state of equilibrium, where the sphere neither floats nor sinks.

However, it must be stated that, if a ball is already dropped with the same density as the liquid itself, it may never be observed neutral buoyancy. Therefore, it would be required,

within the simulation for the user to be able to adapt the spheres' weight to the fluid's own density, while the former is inside the latter.

Simulating the Experiment: Interaction with buoyancy Simulation

It is expected of a user to interact with the experiment, through the GUI developed with *Java Processing* framework. For that to happen, the developed project must be separated into back-end processing and a GUI handler, communicating both through the Processing Framework library.

The simulation basically sums itself up as being similar to an experiment in dropping balls in the ocean, using the ocean floor as a boundary wall in order for them to have a certain dampening factor (in case the drag force, and consequently, buoyancy, is not enough for them to be stopped).

For this simulation, it is important to note the four main components:

1. Ball - a spherical object that falls into the fluid, according to gravitational forces and its drag
2. Water - a fluid of a given density that is responsible for enacting the reaction force of drag, resulting in the buoyancy effect
3. Air - air itself has its own density, acting upon the drag reaction force of the ball
4. Sand - a wall type of object, responsible for serving as a boundary

In this case, two modules have been developed, Physics and Setup, being the former completely responsible for the handling of the forces applied upon the simulation components and the latter being the main loop of this simulation, calling upon a draw method that would be required each milliseconds.

In order to develop the components Water and Air, it has been created a parent class, Fluid, that defines density and how the drag force acts upon objects that interact with such fluids. Drag force is calculated based on the formula on (3). Although Air itself does not need anything specific to be defined, apart from its own density (since it already possesses the function for drag force), Water, being the main actor for the buoyancy simulation, requires a function to calculate the buoyancy of objects. For this, as defined in the formula in (2), it must be known density of the object and its volume. This class also detects if the falling object (the Ball) is inside, in order to apply the necessary forces to it.

Spheres are handled through the class Body, that inherits attributes and methods from Mover.

- **Mover** - defines velocity, acceleration, mass and position in a body (object), defining also how it moves according

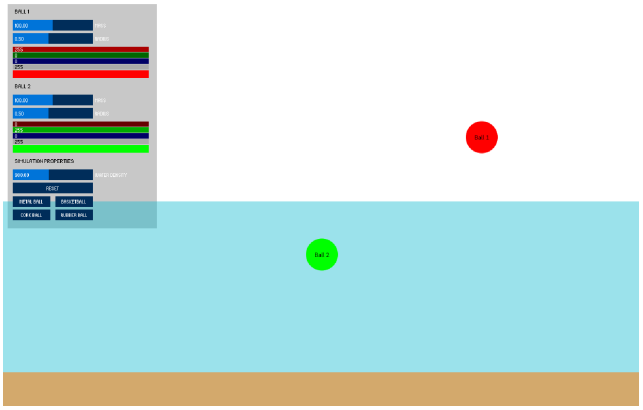

Figure 1: Initial presentation of the buoyancy simulation through a graphical interface, presenting two balls, one that descends from air other that ascends from within the water

to the passage of time

$$\mathbf{vel} = \mathbf{vel} + \mathbf{acc} \cdot dt, \quad (4)$$

$$\mathbf{pos} = \mathbf{pos} + \mathbf{vel} \cdot dt \quad (5)$$

- **Body** - inheritance relationship with Mover, it simply defines how the object should be drawn and if it is colliding with the sand in the bottom of the fluid; it truly is not needed to define anything else regarding the Body class due to it already being provided everything necessary in terms of forces from the Mover class (in a way that enables better modularity)

The Wall class has been defined as a simple boundary within the bottom of the liquid, having the Body class handle collision.

Graphical Simulation

Graphical simulation is handled in two different ways, complementing each other:

- The class itself of a component handles its own drawing
- The app handler class draws, according to the time defined in the setup class, the object each frame, being called upon the method of themselves

When the simulation begins, there are two balls: one that falls down from the air and another that emerges from the middle of the water to its surface. These two balls are completely editable when it comes to **radius, weight** and **color**. Such edits can be done through the use of sliders that edit the former properties.

The simulation itself relies on the fluid density. For that to be editable, there is also present a slider.

For the user to be able to observe how different materials could interact with a liquid of a given density, it has been defined four different pre-built balls, being able to discern how different materials would be affected within fluid density - metal, cork, rubber and basketball.

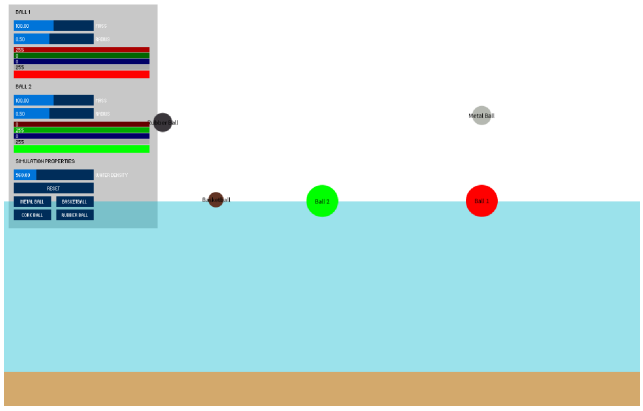

Figure 2: buoyancy simulation making use of two initial balls with some extra pre-built ball types for easier understanding

In order to achieve such within the simulation, two very important code functions have been defined, in order to simulate the way buoyancy, alongside drag force, works.

For the drag force to be applied, its formula has been coded the following way:

```
public PVector drag(Body b) {
    float area = PApplet.pow(b.
        radius, 2.0f) * PApplet.PI;
    float mag = b.vel.mag();
    return PVector.mult(b.vel, -0.5
        f * density * area * mag *
        0.47f);
    }
```

When it comes to the way buoyancy is applied, it has been programmed the following way:

```
public PVector buoyantForce(Body b,
    float g) {
    float volume = (4.0f / 3.0f) *
        PApplet.PI * PApplet.pow(b.
        radius, 3.0f);
    float Fb = this.density *
        volume * Math.abs(g);
    return new PVector(0,Fb);
}
```

Results Regarding the Simulation

When it comes to the simulation it has been observed the following, taking also into account the theoretical principles of buoyancy:

1. When it comes to the two balls of the same size, regarding the one that **falls from the air**:
    - If one is too heavy, it will sink into the liquid given it's density being higher than the liquid itself
    - If one is too light, it will slightly bounce back when hitting the liquid for the first time (due to the higher density of the liquid creating a "rubber-band" effect) and stay floating on its surface

2. When it comes to the two balls of the same size, regarding the one that **emerges from the liquid**:
    - If it is too heavy, its density will be higher than the liquid's which means it will sink into the bottom, being stopped by the sand floor
    - If it is too light, its density will be lower than the liquid's which means it will emerge to the surface, where it may bounce initially after completely stepping outside of the surface boundary of the liquid until it stabilizes on the very surface of the latter

3. If the ball is **already inside** the liquid:
    - By changing the density of the liquid or the weight of the ball, one may observe that until a certain parameter is met that allows for the ball to stabilize its movement within the water it may continue to float upwards or downwards, depending on its current velocity

Therefore, for the ball to achieve **neutral buoyancy** the following values have been set for a density of 900 in the liquid (simulating water):

- Radius - 50 cm
- Weight - 471.2 kg
- Volume - 0.5236 m^3
- Density equals the density of the water itself, by being 900 kg/m^3

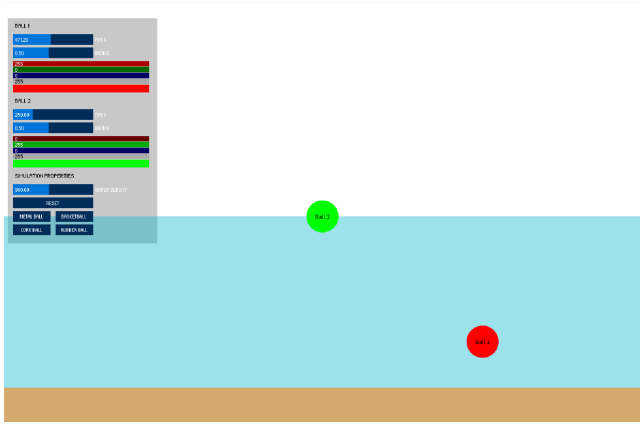

Figure 3: Neutral buoyancy is achieved with the parameters defined; Ball 1 is completely static underwater

When observing the transition into neutral buoyancy, it can be seen that the ball decreases its acceleration until becoming completely static. When densities are alike, an object within a fluid shall become completely still. However, if thrown into the liquid, the acting forces (upon its body) will retain acceleration, with the latter decreasing when in contact with the liquid. In terms of self-organization, it is possible to correlate the former with the observed neutral buoyancy since the latter can be considered a fixed point within the system, a state of equilibrium that formed from a chaotic starting point constantly accelerating until entering in contact with the water. The environment itself is coded for the balls to have their way paved for an end goal of stabilization, be it simply floating on the surface of the water, descending all the way to the bottom or achieving neutral buoyancy (staying in the mentioned fixed point), in a way that would allow for the system to naturally evolve toward a state of minimal energy or maximum stability, since, in this self-organizing process, the chaotic motion of the ball (which would be analogous to the initial state of the system) gradually decreases as it encounters resistance from the fluid.

With these values, a possible combination of neutral buoyancy has been observed, achieving the self-organization goal of the simulation itself.

It has also been observed the following with the pre-built balls:

1. Metal ball
   - Weight - 900 kg
   - Radius - 50 cm
   - Observations - the ball itself should sink since its density is much higher than the water, being at 1719 kg/m³
2. Cork ball
   - Weight - 1 kg
   - Radius - 10 cm
   - Observations - the ball floats immediately since its density is much lower than the water itself at 238.73 kg/m³
3. Rubber ball
   - Weight - 1.246 kg
   - Radius - 14.98 cm
   - Observations - the rubber ball immediately gets propelled since its density is extremely low, having 88.49 kg/m³, achieving a rubber band effect
4. Basketball
   - Weight - 30 kg
   - Radius - 40 cm
   - Observations - the basketball bounces slightly until stabilizing upon the surface of the water, having itself a density of 111.9 kg/m³

Conclusions

In this report we can conclude the objectives have been achieved successfully. Neutral buoyancy is achievable, however the simulation requires some prior knowledge on how buoyancy and drag works since for neutral buoyancy to work, one must fine tune weight and radius according to density. Self-organization is observable in a way that all the balls are able to achieve a fixed point in the system whether it is upon the surface, down below or stopped in the middle of the water (neutral buoyancy). Fine tuning the needed physics was necessary in order for the simulation to work, drag coefficients were not added initially but in turn the drag force did not present good enough hydrodynamics, which means the floating/sinking was not accurate enough. For this reason, the sphere drag coefficient was necessary to be added. Finally, the different balls provide a good sense of how different materials would act upon the physics of the water.

References

[1] Encyclopædia Britannica. Density. *Encyclopædia Britannica* https://www.britannica.com/science/density, 2025.

[2] Encyclopædia Britannica. Specific gravity. *Encyclopædia Britannica* https://www.britannica.com/science/specific-gravity, 2025.

[3] Encyclopædia Britannica. Buoyancy. *Encyclopædia Britannica* https://www.britannica.com/science/buoyancy, 2025.

[4] Encyclopædia Britannica. Drag. *Encyclopædia Britannica* https://www.britannica.com/science/drag, 2025.

[5] NASA Glenn Research Center. Drag of a sphere. Accessed: 2025-01-26 https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/drag-of-a-sphere/.

[6] ScienceDirect Contributors. Self-organization. *ScienceDirect Topics*. Accessed: 2025-01-26 https://www.sciencedirect.com/topics/social-sciences/self-organization#:~:text=Self%2Dorganization%20refers%20to%20spontaneous,likely%20candidates%20for%20self%2Dorganization.

[7] Luís Correia. Introduction to buoyancy and drag, 2025. Lecture Notes from Moodle (Professor's Resource) https://moodle.ciencias.ulisboa.pt/pluginfile.php/549991/mod_resource/content/5/Intro.pdf.

[8] Duarte Gonçalves and Henrique Matos. Project repository, 2025. Java Project developed through Processing Framework for the Artificial Life course at FCUL https://github.com/DuarteGoncalves04/fcul-va-project.

[9] Casey Reas and Ben Fry. Processing: Programming for the media arts https://processing.org/. *AI SOCIETY*, 20:526–538, 09 2006. doi: 10.1007/s00146-006-0050-9.

[10] Study.com Staff. How to calculate the buoyant force of a floating object. Accessed: 2025-01-26 https://study.com/skill/learn/how-to-calculate-the-buoyant-force-of-a-floating-object-explanation.html#:~:text=F%20b%20%3D%20%CF%81%20V%20g,in%20the%20following%20two%20examples!

[11] JoVE Science Education Team. Buoyancy and drag on immersed bodies, 2025. Accessed: 2025-01-26 https://www.jove.com/v/10392/buoyancy-and-drag-on-immersed-bodies#:~:text=Objects%2C%20vehicles%2C%20and%20organisms%20immersed,to%20the%20direction%20of%20motion.