Multi-body collisions in molecular dynamics

Author
Joanna Binek

1. Project objectives

Main objective of this project was to simulate and visualize multi-body collisions in micro scale, it means in molecular dynamics. This project was also about a few scientific experiments on created simulation f.e. test of multi-body collisions' behaviour with and without gravity within the environment.

2. Initial project assumptions

In our project we base on classical Newton's mechanics principles, that means no quantum effects are being considered. We also don't consider friction, so objects do not spin.

3. Selected methods and principles

For our project we choose classical Newton mechanics as the whole simulation is within an inertial frame of reference.
The laws can be described as below:

a) In an inertial of reference, an object either remains at rest or continues to move at a constant velocity, unless acted upon by a force

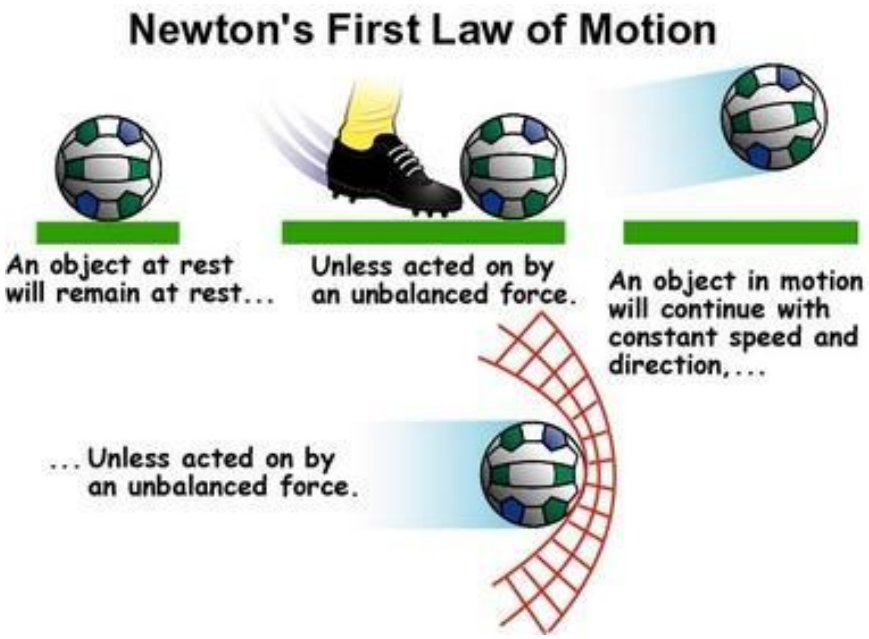

Figure 1 Infographic of Newton's first law of motion

b) In an inertial frame, the vector sum of forces F on an object is equal to the mass *m* of that object multiplied by the acceleration *a* of the object, which can be described by equation: $F = ma$

Where:

- *F*: force
- *m*: mass
- *a*: acceleration

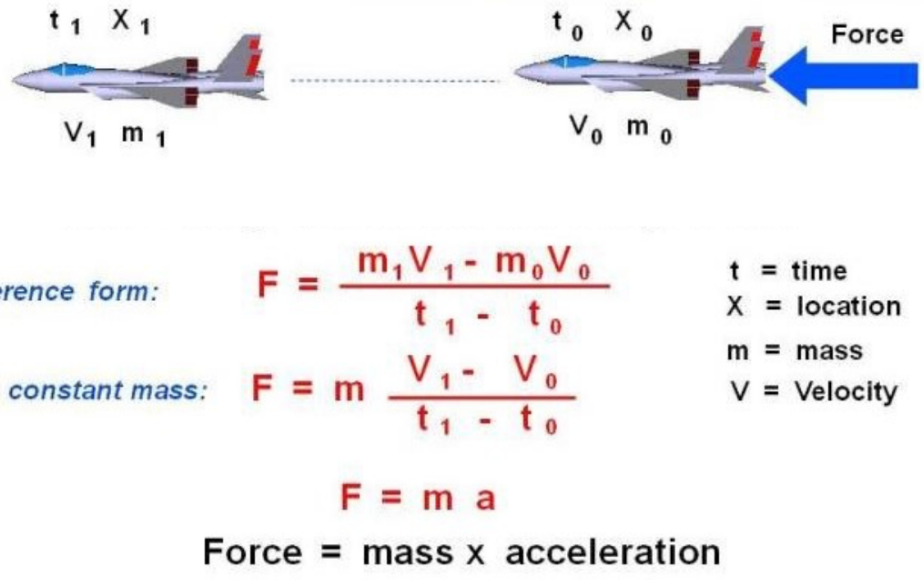

Figure 2 Infographic of Newton's second law of motion

c) When one body exerts a force on a second body, the second body simultaneously exerts a force equal in magnitude and opposite in direction to the first body.

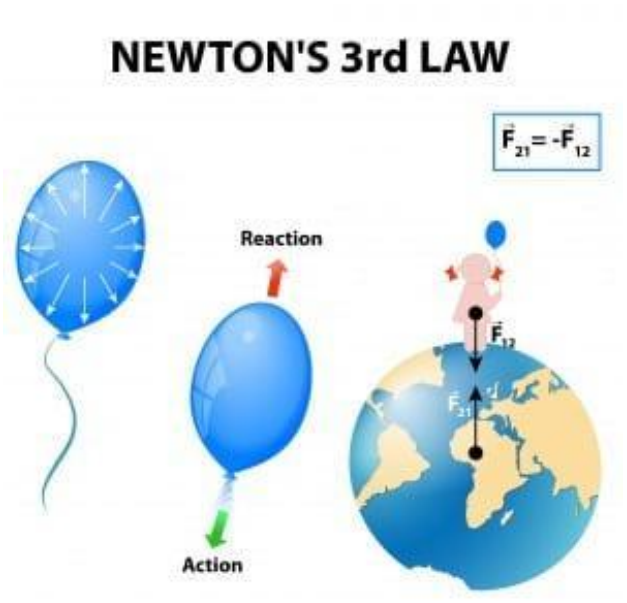

Figure 3 Infographic of Newton's third law of motion

4. Elastic collisions

An elastic collision is a collision in which there is no net loss in kinetic energy in the system as a result of the collision. Both momentum and kinetic energy are conserved quantities in elastic collisions as opposed to inelastic collisions where kinetic energy is lost (converted into heat).
Micro-scale environment is the most preferable environment for elastic collisions because in macro-scale a friction of the object is present. Only the collisions of atoms can fully be considered as elastic collisions.

a) One-dimensional collisions

The simplest form of a collision is one-dimensional collisions. The conservation of the total momentum demands that the total momentum before the collisions is the same as the total momentum after the collision.

Consider two colliding disks with velocities which are not parallel. Lets store these velocities as decomposed vectors in the *x-y plane* (as it is implemented in the code).

Another way to decompose these vectors is along (and normal to) the axis of collision (as in the line going through the center of the two disks). During the collision, the components normal to the axis of collision are unaffected (since we do not consider friction). What we have left is the velocities parallel to the axis of collision. It is recognized as a linear elastic collision!

Applying conservation of momentum we can see that we have one equation with two unknowns:

$$m_A v_{Ai} + m_B v_{Bi} = m_A v_{Af} + m_B v_{Bf}$$

Where:

- v_{Ai}- vector of speed of body 1
- v_{Bi} - vector of speed of body 2
- v_{Af} - vector of speed of body 1 after collision
- v_{Bf} - vector of speed of body 2 after collision
- m_A - mass of body 1
- m_B - mass of body 2

Because kinetic energy is also conserved, we simultaneously have another constraint:

$$\frac{1}{2} m_A v_{Ai}^2 + \frac{1}{2} m_B v_{Bi}^2 = \frac{1}{2} m_A v_{Af}^2 + \frac{1}{2} m_B v_{Bf}^2$$

Having above equations, we can resolve for v_{Af} and for v_{Bf}:

$$v_{Af} = \left(\frac{m_A - m_B}{m_A + m_B} \right) v_{Ai} + \left(\frac{2m_B}{m_A + m_B} \right) v_{Bi}$$

$$v_{Bf} = \left(\frac{2m_A}{m_A + m_B} \right) v_{Ai} + \left(\frac{m_B - m_A}{m_A + m_B} \right) v_{Bi}$$

Case description: $v'_2 = v_1$, because $v_2 = 0$ and both objects have the same mass.

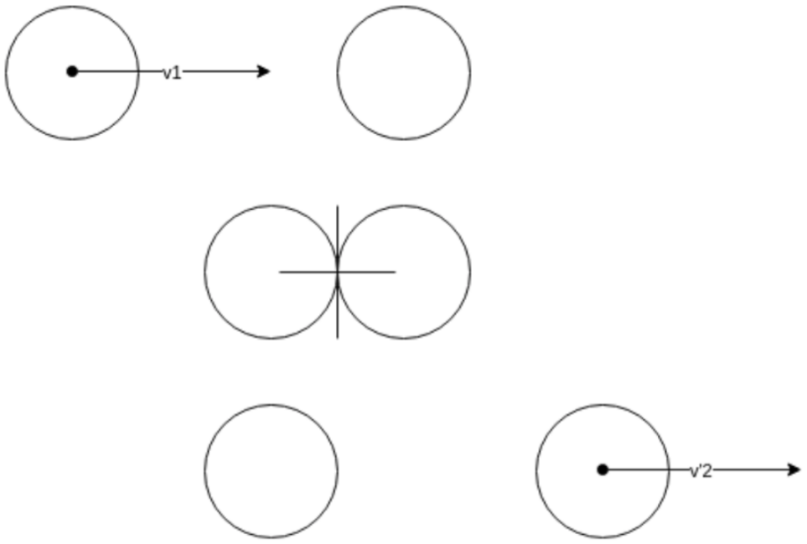

Figure 4 Elastic collisions of two particles, when one particle is moving

Case description: $v'_2 = v_1$ and $v'_1 = v_2$, both objects have the same mass.

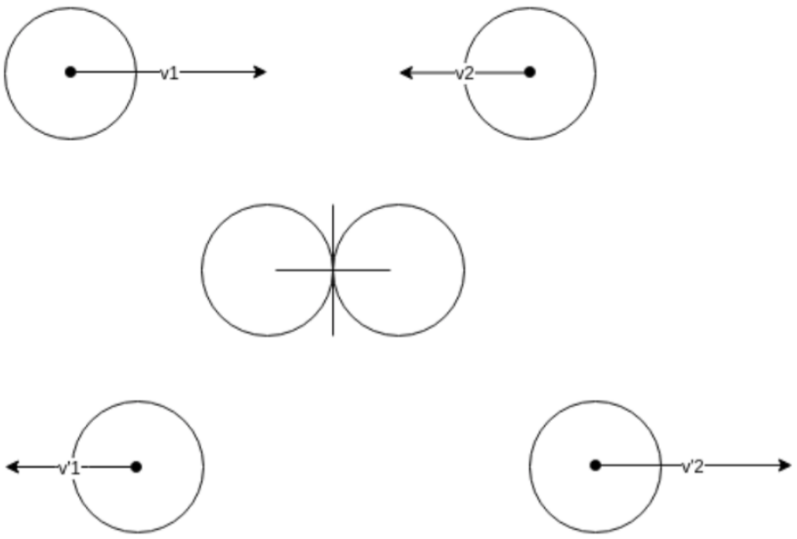

Figure 5 Elastic collisions of two particles, when both particles are moving

b) Two-dimensional collisions

For a collision where objects are moving in two dimensions (e.g. x and y), the momentum will be conserved in each direction independently (as long as there's no external impulse in that direction). In other words, the total momentum in the x direction will be the same before and after the collision.

So, a collision in two dimensions obeys the same rules as a collision in one dimension: Total momentum in each direction is always the same before and after the collision. Total kinetic energy is the same before and after an elastic collision.

Objects' behaviour can be described by below equations:

$$v_1' = v_1 - \frac{2m_2}{m_1+m_2} \frac{\langle v_1 - v_2, x_1 - x_2 \rangle}{||x_1 - x_2||^2} (x_1 - x_2)$$
$$v_2' = v_2 - \frac{2m_1}{m_2+m_1} \frac{\langle v_2 - v_1, x_2 - x_1 \rangle}{||x_2 - x_1||^2} (x_2 - x_1)$$

Where:

- v'_1 - vector of speed of body 1
- v'_2 - vector of speed of body 2
- v_1 - vector of speed of body 1 after collision
- v_2 - vector of speed of body 2 after collision
- m_A - mass of body 1
- m_B - mass of body 2
- x_1 - vector of position of body 1
- x_2 - vector of position of body 2

Case description: $v_1 > 0$, $v_2 = 0$, both objects have the same mass.

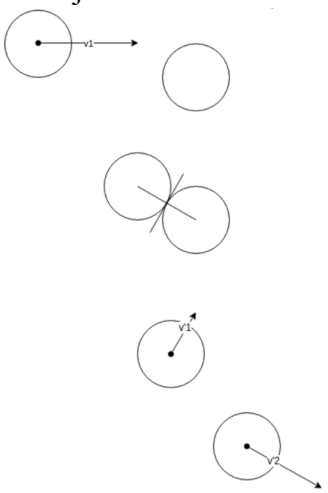

Figure 6 Elastic collisions of two particles in 2D

5. Simulation description

The simulation consists of populating our rectangular simulation space with randomly placed particles. Their masses and velocities are also randomized, within predefined limits set in simulation parameters:

```
circle_r_min, circle_r_max = 0.09, 0.02  # m, m
circle_v_min, circle_v_max = 0.01, 0.2  # m/s, m/s
```

Particles cannot exit the simulation space, they bounce elastically off of the walls. The simulation window can be resized.

All objects are assumed to be equally dense, so if they differ in size then they also differ in mass, which makes it easier to distinguish heavy ones from the lighter ones.

The main logic of the simulation is based on collision - when two objects collide with each other their colors are randomly changed to visualize the act of the collision. Colors of the objects are not changed when they collide with the walls.

We use objects that are alike to hokey disks. They collide with each other, they can have different sizes. We start our simulation with an initial number of 10 disks, then it can be increased up to 100 disks.
Every time the number of objects is changed, a new amount of disks appear in the simulation window with newly and randomly picked positions.

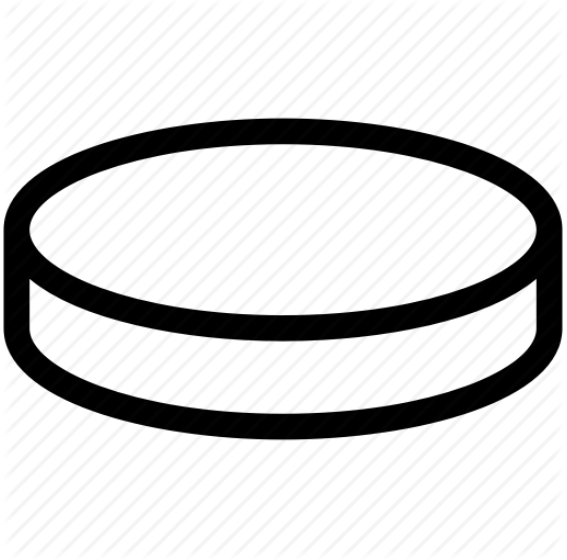

Figure 7 Graphic of object used in the simulation

6. Technical overview

Our simulation code was written in Python language (version 3). We used **pygame** library to simulate multi-body collisions in a rectangular area.
Pygame is a free and open-source cross-platform library for the development of multimedia applications like video games using Python. It includes computer graphics and sound libraries designed to be used with the Python programming language.
We also used numpy, scipy and matplotlib libraries to perform mathematical operations and visualize their effects on charts.

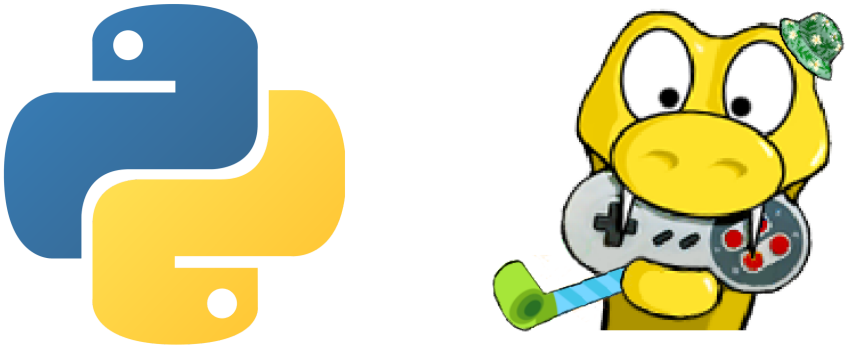

Figure 8 Python and pygame technologies

We used Jupyter Notebook as an environment to run our code. It was convenient for us to split the code for a few blocks (as it is done within the Jupyter) to set the logic of the program in a clear way.

Figure 9 Jupyter Notebook logo

7. GUI description

The Graphical User Interface of our project consists of a rectangular area (1050 x 750) where we simulate multi-body collisions by colliding disks.

As it was said before - the main logic of the simulation is based on collision - when two objects collide with each other their colors are randomly changed to visualize the act of the collision.

The simulation starts with an initial number of objects - 10 and it can be increased by the user up to 100 objects. Users can increase this number by pressing UP ARROW and analogically, they can decrease the number of objects by pressing DOWN ARROW.
And as it was also previously mentioned - every time the number of objects is changed, a new amount of disks appear in the simulation window with newly and randomly picked positions.

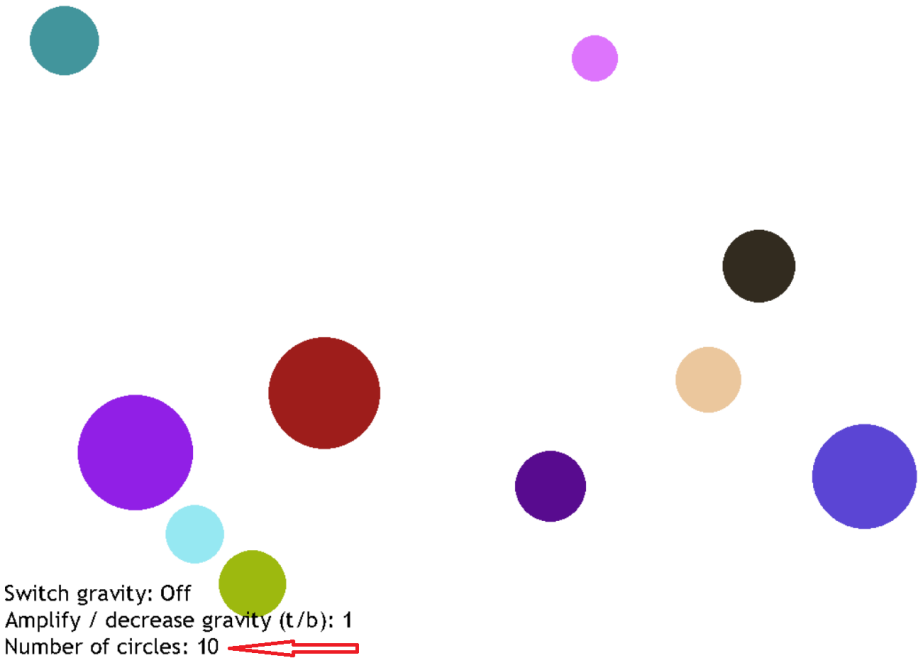

Figure 10 Simulation window with initial number of objects

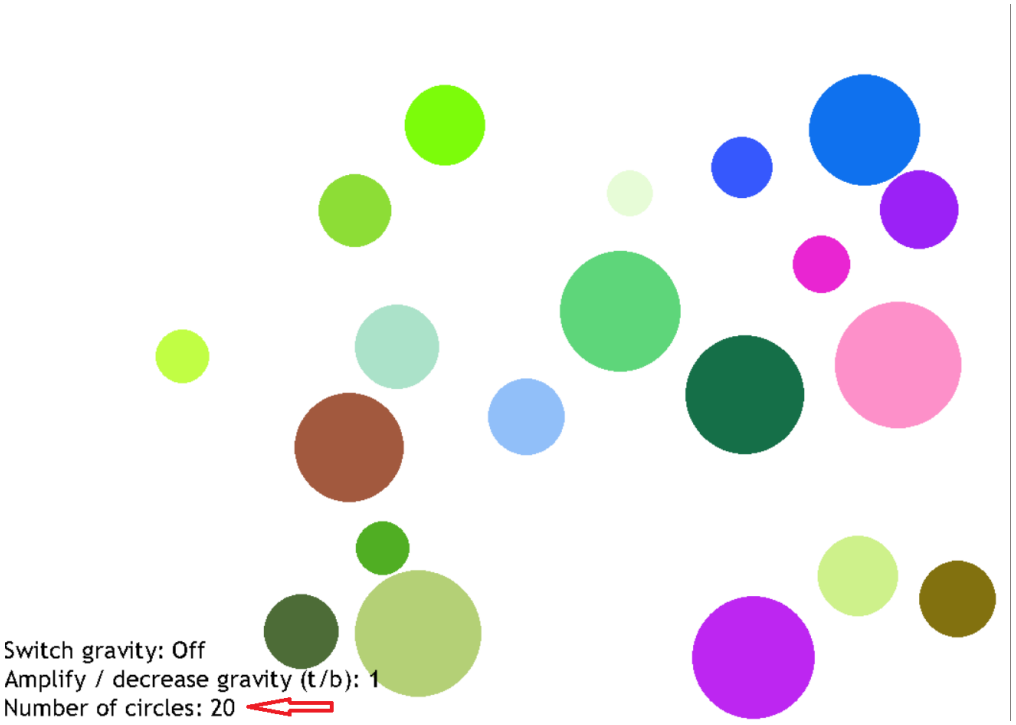

Figure 11 Simulation window with modified number of objects (20)

Another option for the user in our GUI is the possibility to “turn on” and “turn off” the gravity within the environment. Initially the gravity is not considered in the simulation, but the user can turn it on by pressing the G button and then turn it off by pressing G button one more time.
It is also possible to increase and decrease the value of gravity by pressing the T or B button.

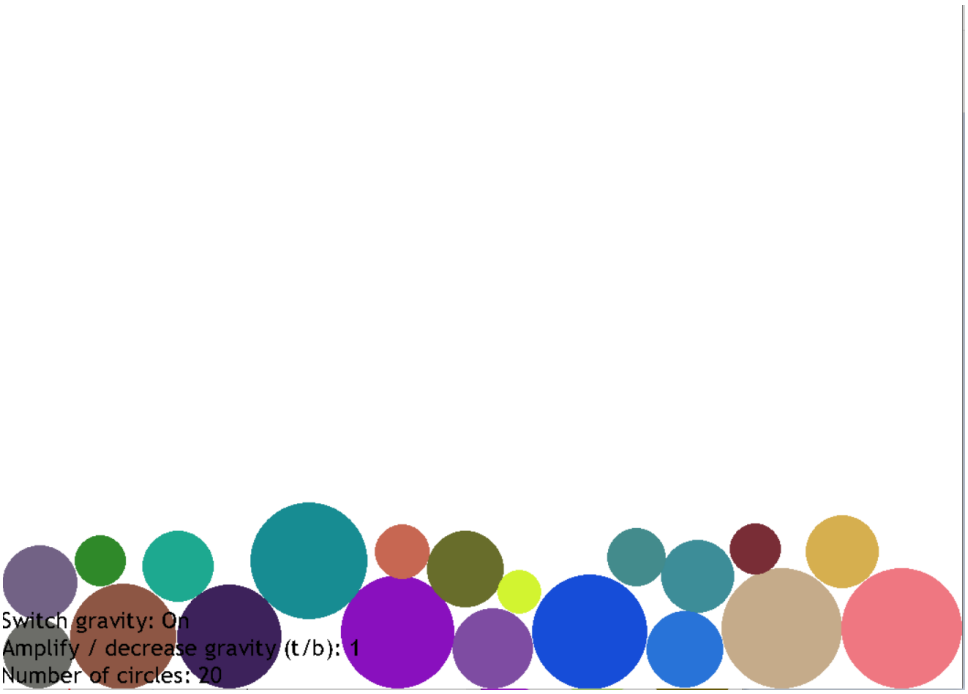

Figure 12 Simulation window with gravity set to “on” after some time

Last but not least, the user can perform a click on the simulation window and make all existing objects move to the place of the click immediately.

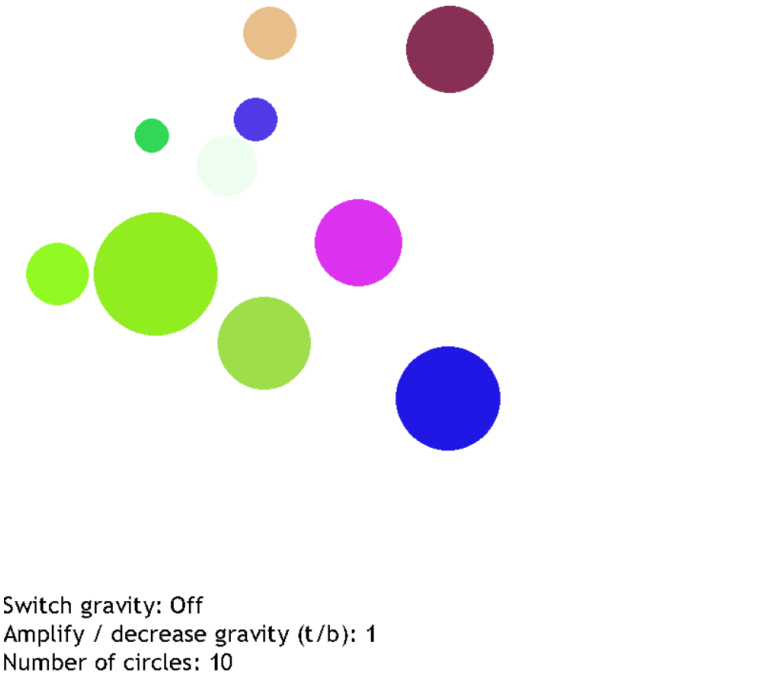

Figure 13 Simulation window after clicking the area

The simulation widow can be loosely resized.

8. Conducted experiments

a) Presence and absence of the gravitation within the environment

As it was previously mentioned, the user has the possibility to "turn on" and "turn off" the gravity within the environment. Initially the gravity is not considered in the simulation, but the user can turn it on by pressing the G button and then turn it off by pressing G button one more time.
It is also possible to increase and decrease the value of gravity by pressing the T or B button. It causes multiplication of the default value of the gravity (9.80665 m s^-2) by the set value. The implementation looks like below:

```
self.v = self.v + np.array([0, g_amp * g]) * time_d * px_per_m
```

Where:

- selv.v - velocity of an object
- time_d - current time of the simulation
- px_per_m - pixel per meter, distance that each object travels
- g - standard acceleration of gravity imported from scipy.constats (9.80665 m s^-2)
- g_amp - gravity amplifier set by the user

Thanks to that we can observe the situation when the gravitation is higher or even lower than usual. We can model a case when the gravitation has negative value and we can see the simulation “upside down”.

The most interesting part about it is the transition moment between positive and negative value of the gravitation, because we can see the exact movement of all objects from the bottom of the simulation window to its ceiling.

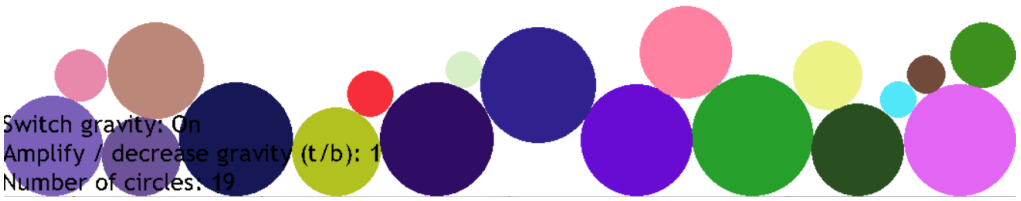

Figure 13 Simulation window with gravity set to “on” with value = 1

In the picture below we can observe the “transition moment” when there is no gravity again and the objects start slowly floating.

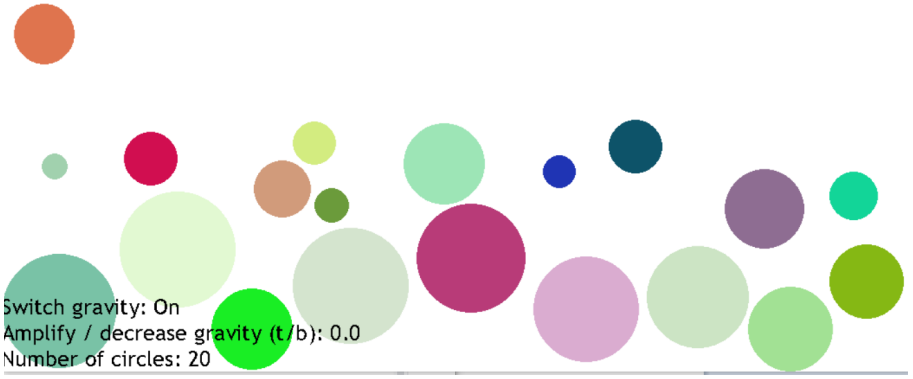

Figure 14 Simulation window with gravity set to “on” with value = 0.0

After setting a negative value of the gravity we can observe the “upside down” situation in the simulation window. Now the ceiling is the new ground.

Switch gravity: On
Amplify / decrease gravity (t/b): -1.0
Number of circles: 19

Figure 15 Simulation window with gravity set to “on” with value = -1.0

b) Simulation of the magnet phenomena

As it was mentioned previously, the user can perform a click on the simulation window and make all existing objects move to the place of the click immediately.

We called it a “magnet phenomena”, because we wanted to simulate the situation, when all objects are made of metal and the click symbolizes the existence of a magnet in that place. It makes all the objects move toward the “click place”, just like they were attracted by the magnet.

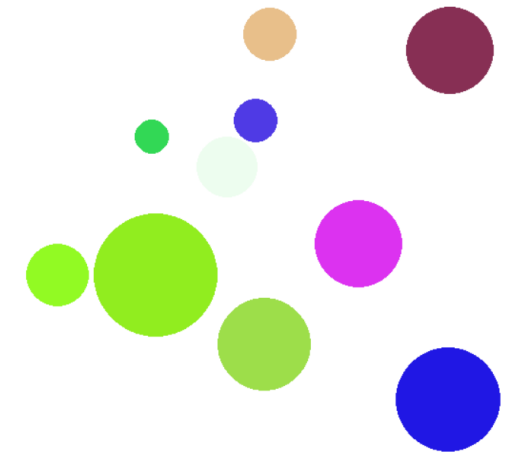

Figure 16 Simulation window with the magnet phenomena

c) Measurement of the number of collisions in time and after switching on the gravity

With this experiment we wanted to visualize on a chart how the number of collisions is changing in time and how it is changing after turning on the gravitation functionality.

We decided to mark the “switch on” moment with the red vertical line on the chart.

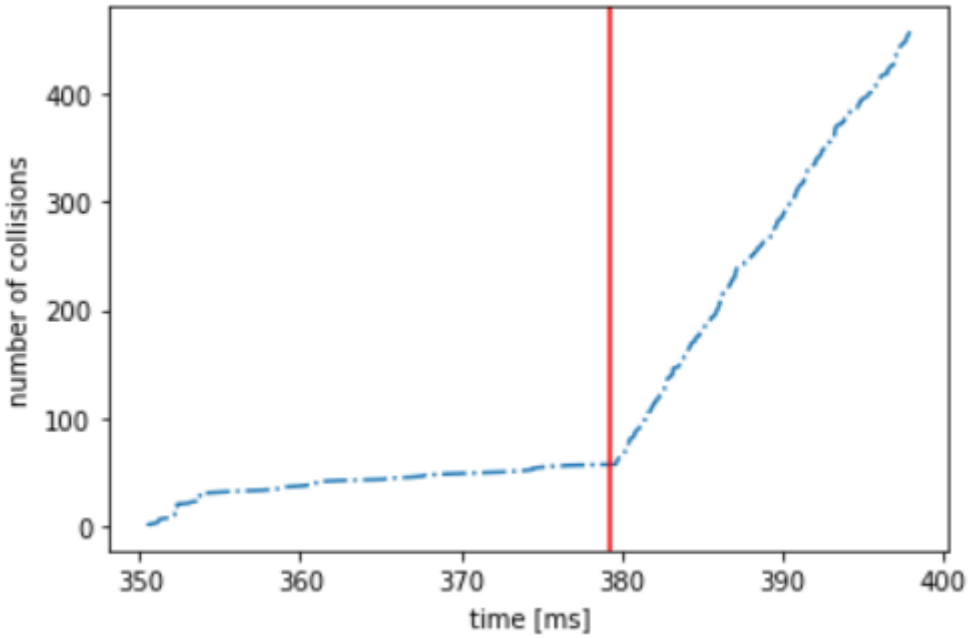

Figure 17 Number of collisions in time and after turning on the gravity

We can observe that during the simulation that lasted for 400 milliseconds (when turning on the gravity occured around 378th millisecond) the number of collisions increased from ~50 to ~400 only after ~20 milliseconds.

It was caused by the impact of gravity, as it makes all objects fall down to the ground direction which causes many more collisions than when they’re just floating.

9. Future improvements

Application created by us to perform the above experiments is rather simple, but it satisfies all initial requirements, so the whole project can be recognized as completed.

However we see fields that can be improved in the future.

The GUI itself can have a more “modern” format or can be displayed more separately from the simulation window which can improve the whole view of the application.

There is a small problem during resizing the window. When it happens, the simulation visibly stops. It does not impact the whole effect of the simulation as after resizing everything works fine, but it is something that can be improved.

Last but not least, it would be nice to propagate this application to the Internet, f.e. via some cloud provider like AWS or Google Cloud to expand the audience.