Alexey Elkin

Investigating the Rate of Protostar Formation with an **N-Body Simulation**

Alexey Elkin

Investigating the rate of Protostar formation with an N-body Simulation

The purpose of this investigation is to model the coalescence of particles forming a Protostar and to find factors affecting the rate of coalescence. This will be done using an "N-Body Simulation", a computer program that models the interactions of particles while under the influence of gravity. I wrote the program in *Java*, using *JavaFX* to create the GUI. The program code and development timeline are all hosted on *Github*; there is a link at the end of the document.

In the N-body simulation, each particle has the following attributes: **mass**, **colour**, **location**, **velocity**, and **temperature** (although the latter isn't relevant in the investigation). **Mass** and **temperature** are stored as floating-point numbers, but **location** and **velocity** are stored as 2D Vectors. All the other properties of the particle (such as the radius) are derived from these attributes.

The relationship between the rate of coalescence and 4 factors is analysed. These factors are: **particle density**, **particle mass**, **initial particle speed**, and **particle distribution shape**.

```
private double mass;
private Color color;
private Point2D location;
private Point2D velocity;
private float temperature;

public Particle(double mass, Color color, Point2D location) {
    this.mass = mass;
    this.color = color;
    this.location = location;
    this.velocity = new Point2D(0, 0);
    this.temperature = 0; // The initial temperature of each particle is zero
}
```

The Physics Engine

$$F = G\frac{m_1 m_b}{r^2}$$

"Newton's Law of Universal Gravitation states that every particle attracts every other particle in the universe with a force which is directly proportional to the product of their masses and inversely proportional to the square of the distance between their centres" *(Wikipedia Contributors, 2020a)*

```
if(p.getParticle(i) != p.getParticle(j)) {

    force = p.getGravConstant() * (p.getParticle(i).getMass() * p.getParticle(j).getMass() /
            (Math.pow(p.getParticle(j).getLocation().distance(p.getParticle(i).getLocation()), 2)));

    double alpha = Math.atan2(xDifference, yDifference);
    double angle = Math.toDegrees(alpha);

    xF = force * Math.sin(alpha);
    yF = force * Math.cos(alpha);

    p.getParticle(j).accelerate(new Point2D(xF/p.getParticle(j).getMass(), yF/p.getParticle(j).getMass()));


}
```

The gravitational force acting on the particles is initially calculated using Newton's law of universal gravitation. The angle between each particle in series is then calculated using the **Java.Math atan2()** function, which returns the angle between the particles in radians. The total force acting on the particle is then resolved for each component using the **cosine rule**. The variables **xF** and **yF** represent the X and Y components of the gravitational force.

The gravitational force acting on a particle is calculated by multiplying the product of the masses of the two particles by the gravitational constant, and then dividing by the square of the distance between the particles.

Alexey Elkin

$$F = ma$$

"Newton's Second Law - In an inertial frame of reference, the vector sum of the forces **F** on an object is equal to the mass *m* of that object multiplied by the acceleration **a** of the object: **F** = *m***a**. (It is assumed here that the mass *m* is constant) *(Wikipedia Contributors, 2020b)*

```
p.getParticle(j).accelerate(new Point2D(xF/p.getParticle(j).getMass(), yF/p.getParticle(j).getMass()));
```

Finally, the particle is accelerated following **Newton's 2nd Law**. The acceleration is equal to the **Force / Mass**

$$m_a u_a + m_b u_b = m_t u_t$$

$$m_1 v_{1,i} + m_2 v_{2,i} = (m_1 + m_2) v_f$$

A **Perfectly Inelastic Collision** occurs when the maximum amount of kinetic energy of a system is lost. In a perfectly inelastic collision, i.e., a zero coefficient of restitution, the colliding particles stick together. In such a collision, kinetic energy is lost by bonding the two bodies together *(Wikipedia Contributors, 2020c)*

```
if ((p.getParticle(j).getLocation().distance(p.getParticle(i).getLocation()))
        < (p.getParticle(j).getRealDimensions() / 2 + p.getParticle(i).getRealDimensions() / 2)
        && p.getParticle(i) != p.getParticle(j)) {

    if (p.getParticle(j).getMass() > p.getParticle(i).getMass()) {
        //Larger particle changes its trajectory according to Newton's Third Law
        p.getParticle(j).setVelocity(
                new Point2D(
                        (p.getParticle(j).getMass() * p.getParticle(j).getVelocity().getX() +
                                p.getParticle(i).getMass() * p.getParticle(i).getVelocity().getX()) /
                                (p.getParticle(j).getMass() + p.getParticle(i).getMass()),

                        (p.getParticle(j).getMass() * p.getParticle(j).getVelocity().getY() +
                                p.getParticle(i).getMass() * p.getParticle(i).getVelocity().getY()) /
                                (p.getParticle(j).getMass() + p.getParticle(i).getMass())
                )
        );
        //Larger particle absorbs smaller particle
        double initialKineticEnergy = p.getParticle(i).getKE() + p.getParticle(j).getKE();

        p.getParticle(j).addMass(p.getParticle(i).getMass());

        double finalKineticEnergy = p.getParticle(j).getKE();
        p.getParticle(j).incTemperature((initialKineticEnergy-finalKineticEnergy)/(p.getParticle(j).getMass()*1000));

        p.destroyParticle(i);

    } else {

        //Larger particle changes its trajectory according to Newton's Third Law
        p.getParticle(i).setVelocity(
                new Point2D(
                        (p.getParticle(j).getMass() * p.getParticle(j).getVelocity().getX() +
                                p.getParticle(i).getMass() * p.getParticle(i).getVelocity().getX()) /
                                (p.getParticle(i).getMass() + p.getParticle(j).getMass()),

                        (p.getParticle(j).getMass() * p.getParticle(j).getVelocity().getY() +
                                p.getParticle(i).getMass() * p.getParticle(i).getVelocity().getY()) /
                                (p.getParticle(i).getMass() + p.getParticle(j).getMass())
                )
        );
        //Larger particle absorbs smaller particle
        double initialKineticEnergy = p.getParticle(i).getKE() + p.getParticle(j).getKE();

        p.getParticle(i).addMass(p.getParticle(j).getMass());

        double finalKineticEnergy = p.getParticle(i).getKE();
        p.getParticle(i).incTemperature((initialKineticEnergy-finalKineticEnergy)/(p.getParticle(i).getMass()*1000));

        p.destroyParticle(j);
    }
}
```

Alexey Elkin

The collision detection is done by calculating the distance between two particles and comparing it to the radii of the two particles. If the distance between two particles is smaller than the sum of their radii, a collision has occurred. Once this is established, the larger particle "absorbs" the smaller particle in a perfectly inelastic collision. (The mass of the smaller particle is added to the mass of the larger particle, and the smaller particle is deleted)

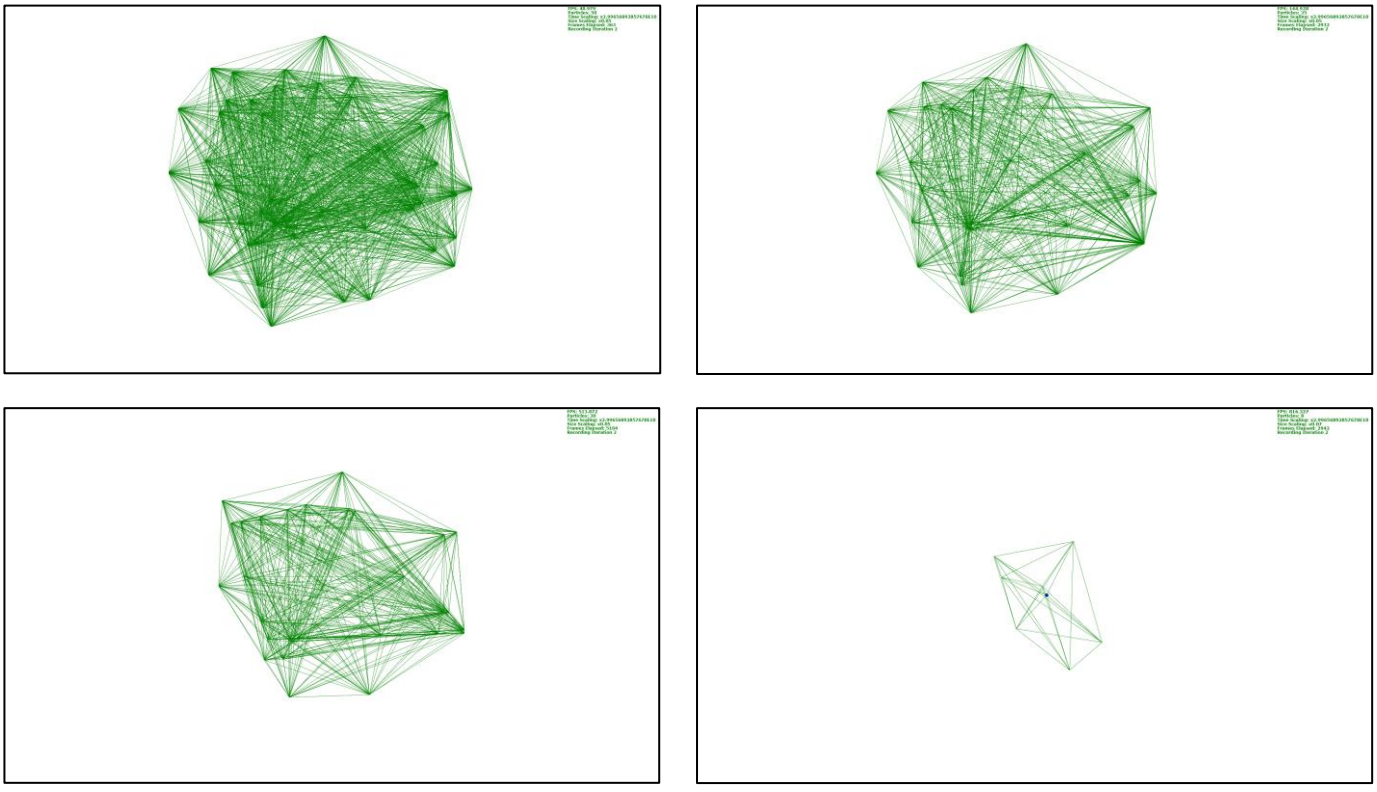
After the particles have coalesced, the **velocity** of the larger particle changes, because it has been altered by the impact of the smaller particle in accordance to **Newton's 1st Law**. The new velocity of the larger particle is calculated by adding the product of its mass and velocity to the product of the mass of the smaller particle and its respective velocity, and then dividing by the sum of the masses of the two particles. This calculation is performed twice – once for the X and Y components of velocity.

The **Kinetic Energy** lost is then converted into **Thermal Energy**. The conversion is done with $\mathbf{Q = m \cdot c \cdot \Delta T}$, but the heat capacity of each particle is assumed to be **1**, so it is ignored in the calculation.

The Experiments

Control Experiment

Particle Number	Particle Mass (Kg)	Initial Speed (ms^{-1})	Particle Distribution	Frames Recorded
100	**2-200**	**0**	**Circular**	**100,000 (x100)**



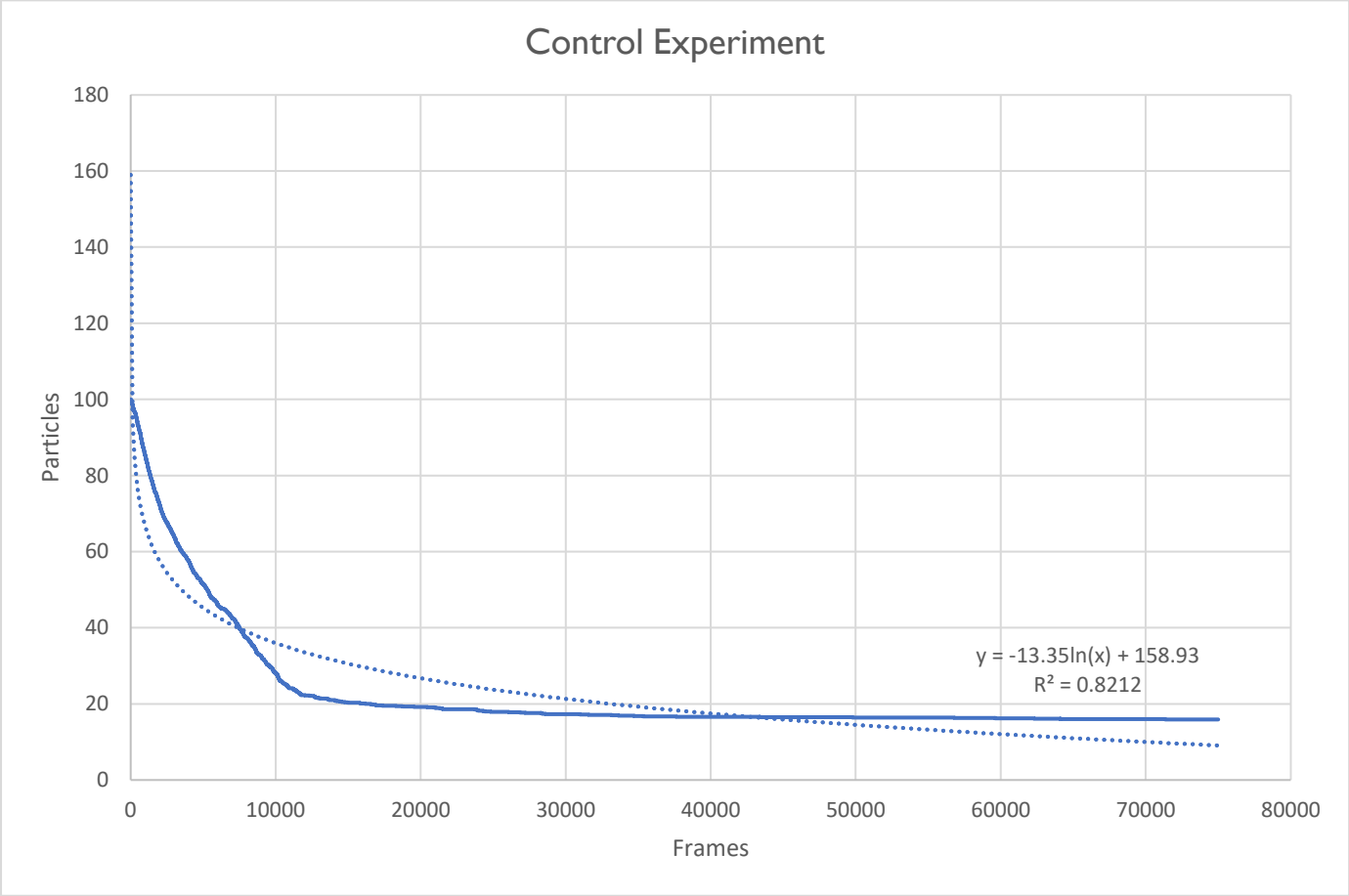
The pictures above show the particles converging to the centre and coalescing as time progresses. The pictures are ordered from left to right, and from top to bottom. The green mesh represents the forces of gravity acting between the particles. It is useful in helping to visualize the approximate distribution of the particles at a specific time in the experiment.

In this experiment, default control conditions for the following experiments were established. The experiment was repeated 100 times, and mean values were taken for the data points to smooth out the curve on the graph and reduce the influence of anomalous data points. A graph of **particle number** VS **frames was recorded**. The particle simulation was assumed to be a perfectly isolated system with no external forces acting on the particles.

Alexey Elkin

In all the experiments, 1 frame corresponds to 14,983,338,528 seconds, or approximately 173,418 years. This means that for an experiment lasting 100,000 frames, about 18 billion years would have elapsed. However, the vast majority of the particles coalesce by about 10,000 frames.

The speed of this coalescence is what I intend to investigate.



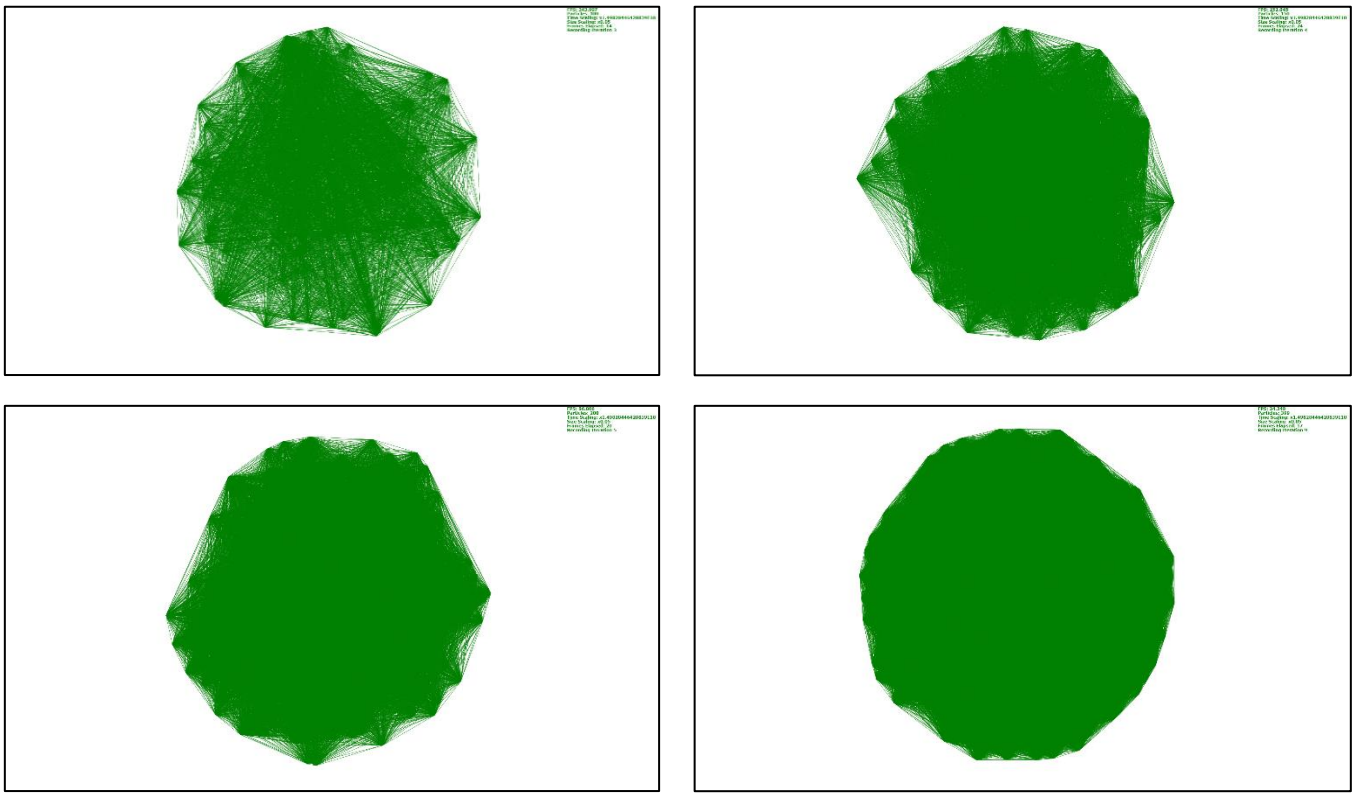
As the data above demonstrates, the rate of particle collisions decreases as the simulation progresses. The relationship between the number of particles and time (frames) can be modelled relatively well with a logarithmic equation: ***-13.35ln(x) + 158.93*** , with **R^2 = 0.82** indicating strong correlation.

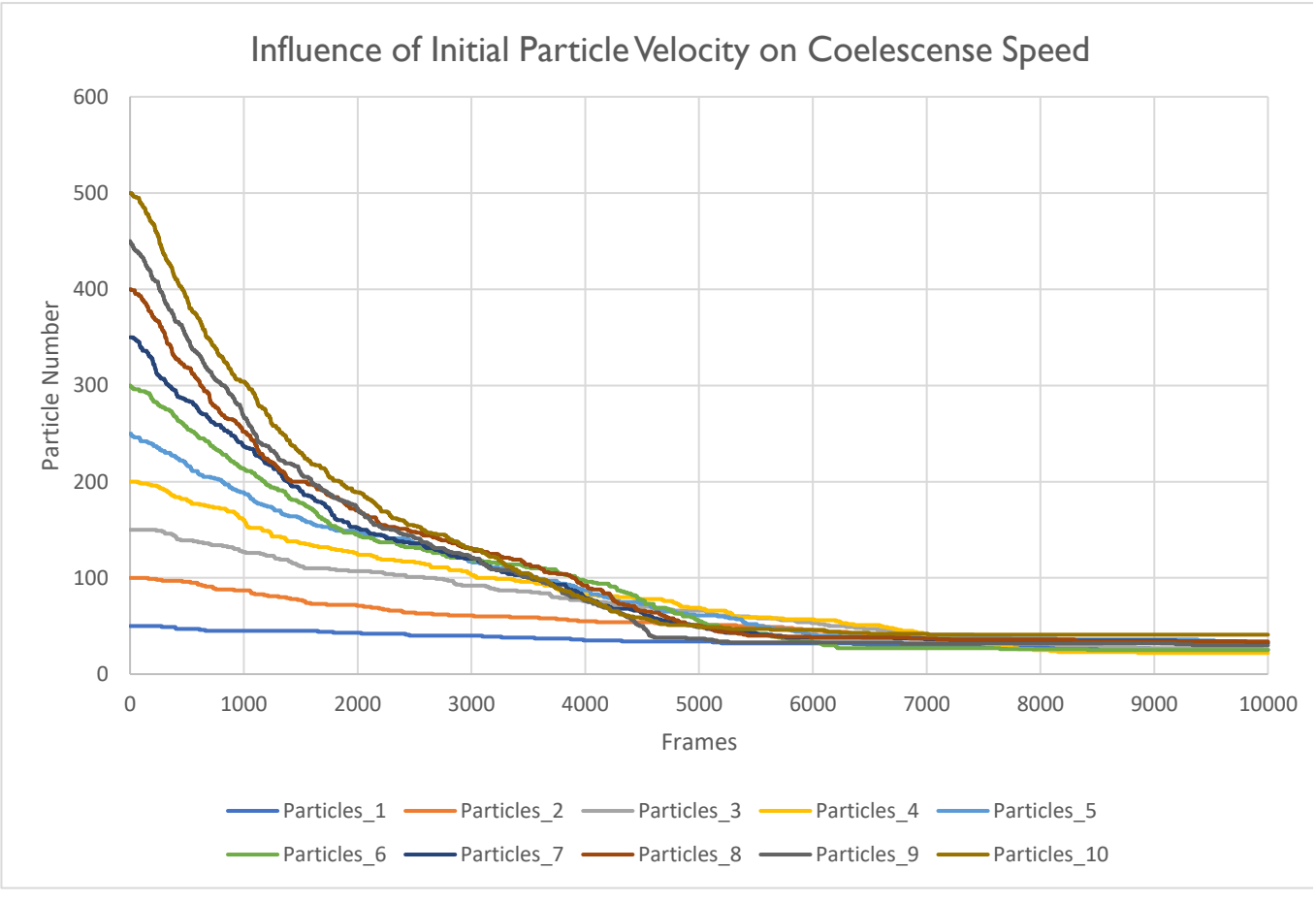
In the experiment, the particles converged towards the centre of gravity at an increasing velocity. As they approached closer to the centre, they coalesced to form one very large particle. Some of the particles which were generated at the far edges of the distribution missed the large mass in the centre, travelling past it. However, with time, they began to return and impact it. By 75,000 frames, there were still about 18 particles remaining. These particles have not coalesced because they have missed other particles on their first approach to the centre and were ejected far off into space. Extrapolation of the data suggest that these particles could take up to 150,000 frames to return to the centre.

Alexey Elkin

Influence of Particle Distribution Density on Coalescence Speed

Particle Number	Particle Mass (Kg)	Initial Speed (ms^{-1})	Particle Distribution	Frames Recorded
100-500 (+50 / inc)	**2-200**	**0**	**Circular**	**100,000 (x10)**





Alexey Elkin

The pictures on the left show the start of 4 consecutive experiments. Initially, the particle density is relatively low. However, further experiments have a much larger particle density – highlighting the circular distribution of the particles.

The purpose of this experiment was to find out how particle density affects the speed at which the particles coalesce. The experiment was performed 10 times, each time with an increasing number of particles (in steps of +50). The first experiment had 50 particles, the second – 100, the third – 150, and so on.

The graph demonstrates that there is a strong correlation and a causal relationship between the initial number of particles and the time it takes for them to coalesce. It can clearly be seen that the curve with the most initial particles decreases the quickest. After about 8000 frames most of the particles have coalesced, and all the curves flatlined. Although the number of particles was recorded to 100,000 frames, the graph only shows 10,000. This was done because the curves remain almost stationary for the next 90,000 frames.

Hypothesis: *There is a positive correlation between the particle density and the rate of coalescence.*

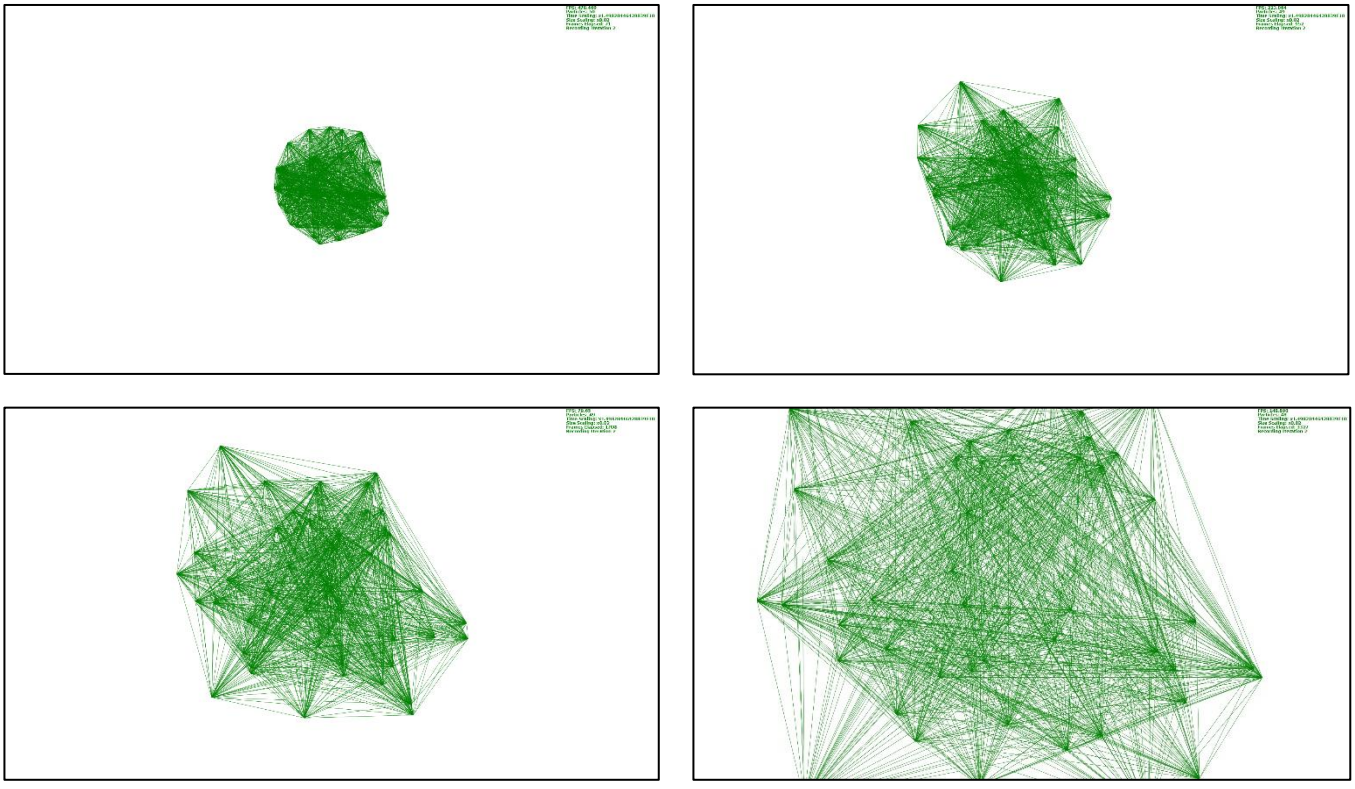
Conclusion: *There is a very strong positive correlation and causal relationship between the particle density and the rate of particle coalescence. (Hypothesis confirmed)*

It should be noted that in this experiment CPU load increased dramatically as the number of particles in the simulation increased. When the number of particles was 50, the average FPS was around 2000. However, as the number of particles increased, the FPS dropped to less than 1 frame per second. This is due to the algorithmic complexity being $O(n)^2$ - *If the number of particles in the simulation doubles – the CPU load quadruples, etc...*

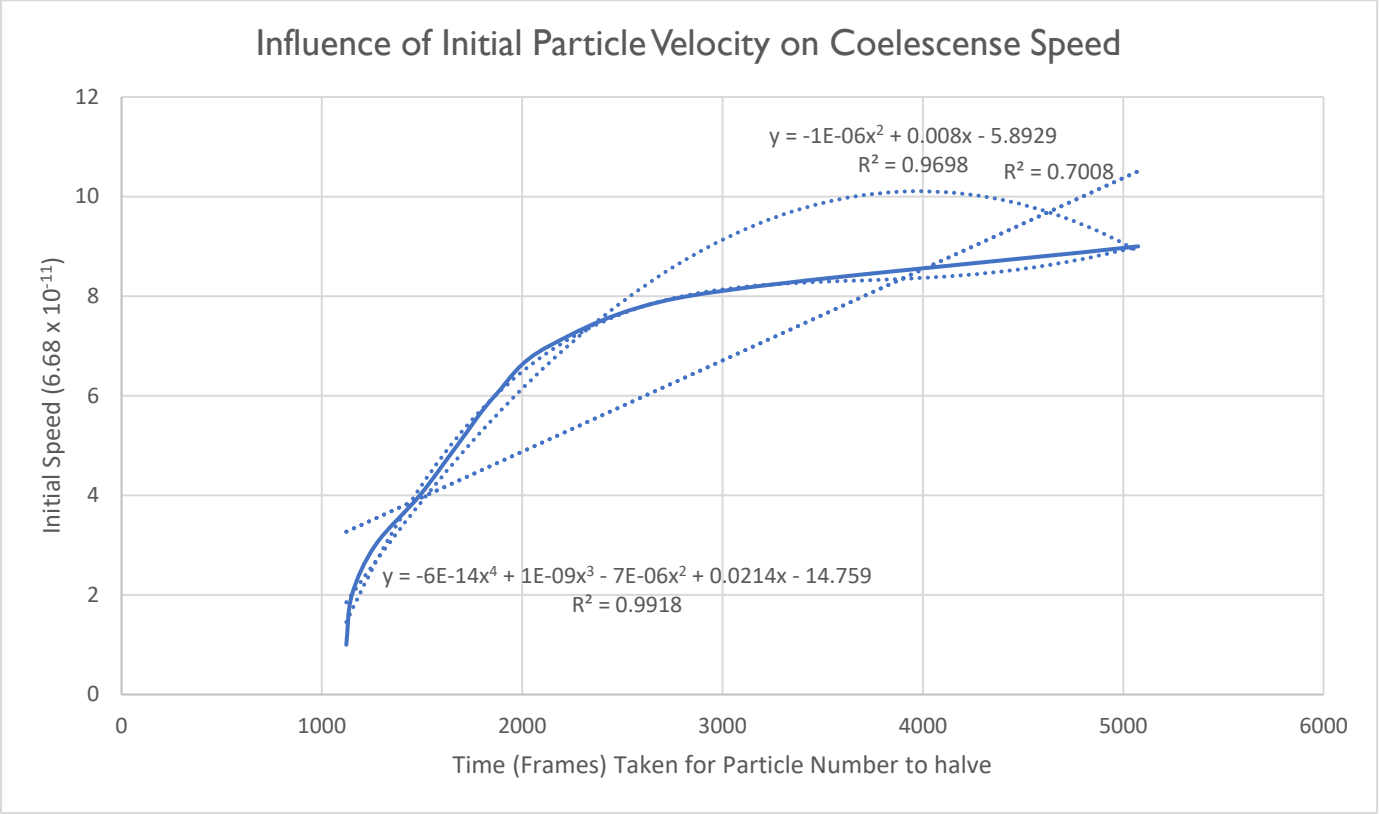
Therefore, it was unsurprising that this experiment took the longest to run - sixteen and a half hours.

Influence of Initial Particle Speed on Coalescence Speed

Particle Number	Particle Mass (Kg)	Initial Speed (ms^{-1})	Particle Distribution	Frames Recorded
100	**2-200**	**6.68E-11 - 6.68E-10**	**Circular**	**100,000 (x10)**



Alexey Elkin



The images show the particles spreading out from the centre of the screen, as their initial velocity carries them further and further from the centre of mass.

The purpose of this experiment was to find out how the initial speed of the particles affected the rate at which they coalesce. The simulation was run in conditions identical to those in the control experiment, the only difference being that the particles had a non-zero initial velocity.

The graph above demonstrates that there is a relatively strong negative correlation between the initial particle speed and the rate of coalescence, $\mathbf{R^2=0.7}$. The graph can be approximated with a curve of best fit with equation $\mathbf{y = 10^{-6}x + 0.008x - 5.8929}$. However, and even better approximation is $\mathbf{y = -6E\text{-}14x^4 + 1E\text{-}09x^3 - 7E\text{-}06x^2 + 0.0214x - 14.759}$

The graph shows that particles with a higher initial velocity take longer to coalesce. This can be explained by the fact that collisions are less likely when the particles start with an initial velocity. The gravitational force acting on the particles decreases because the density of the particle distribution decreases throughout the experiment - the particles spread out as their initial velocity pushes them progressively further away from each other, and the centre of gravity. The images captured in this experiment demonstrate the way that the particles spread out, filling the entire screen.

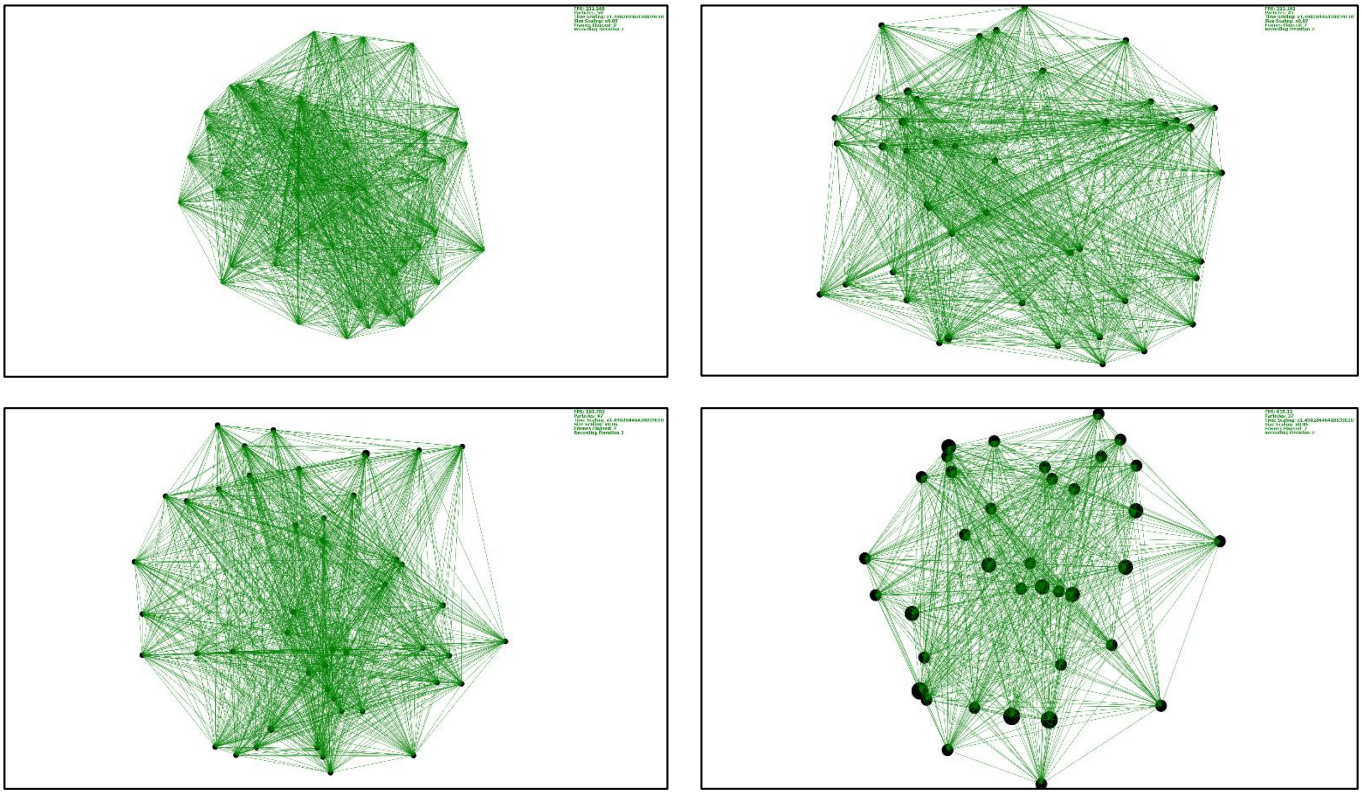
Hypothesis: *There is a positive correlation between the initial particle velocity and the time required for the number of particles in the simulation to halve.*

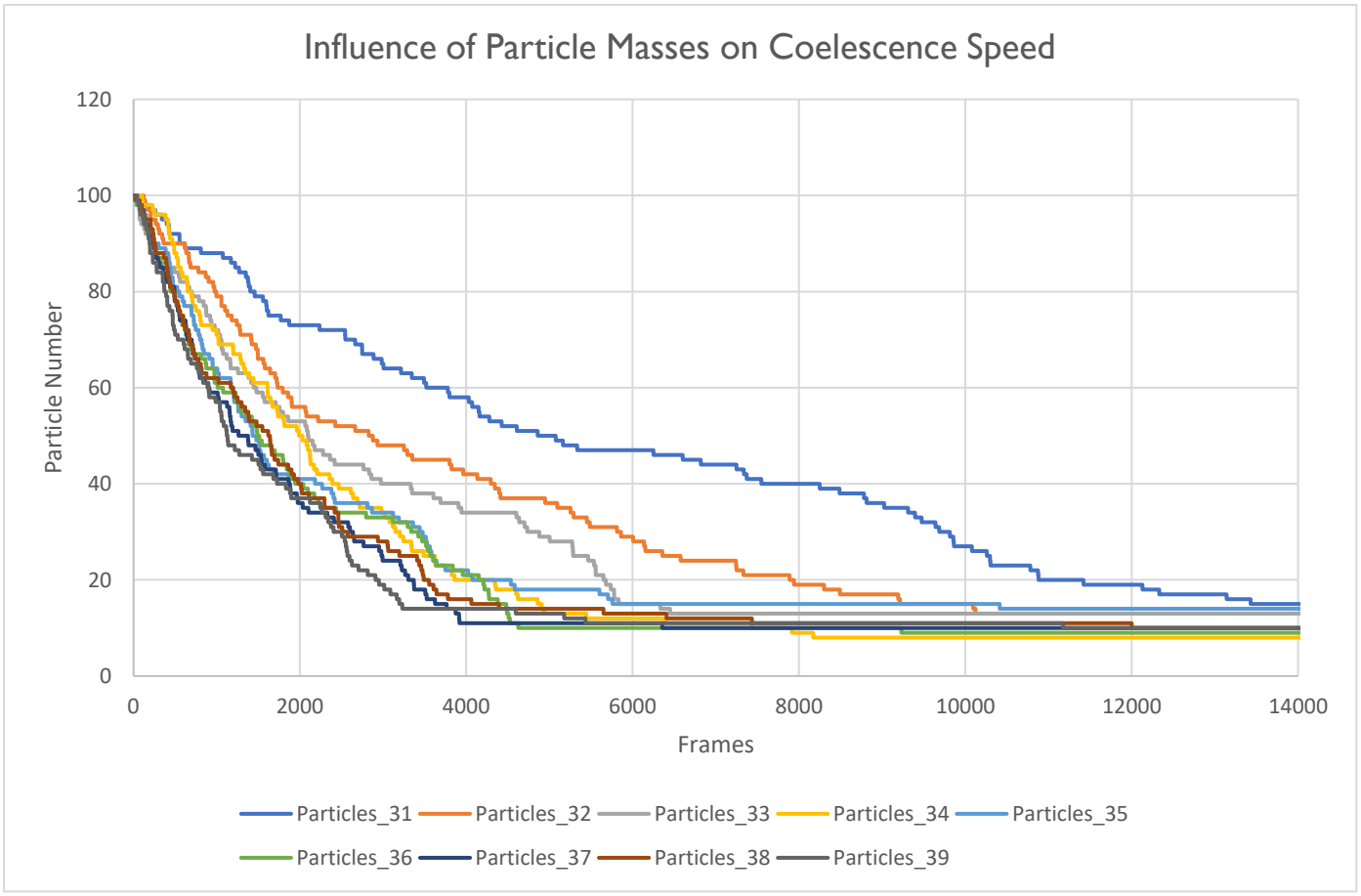
Conclusion: *There is a positive correlation between the initial particle velocity and the time required for the number of particles in the simulation to halve (Hypothesis confirmed)*

Alexey Elkin

Influence of Particle Mass on Coalescence Speed

Particle Number	Particle Mass (Kg)	Initial Speed (ms^{-1})	Particle Distribution	Frames Recorded
100	**100-1000**	**0**	**Circular**	**50,000 (x9)**





Alexey Elkin

The pictures show the simulation running with particles of different masses.

The purpose of this experiment is to find out the relationship between the mass of the particles and the rate at which they coalesce. The experiment was run 10 times, with increasing particle masses (from 10-1000kg, with increments of 100kg)

The graph demonstrates that there is a clear positive correlation between the mass of the particles and the rate at which they coalesce – as the mass of the particles increases, the time taken for them to coalesce decreases.

This is because Newton's Law of Universal Gravitation states that the gravitational force between two particles can be represented with $F = G\frac{m_1 m_1}{r^2}$. Since mass of the particles is directly proportional to the gravitational force acting on them, larger particles will experience a larger gravitational force acting on them.

However, Newton's Second law states that $F = ma$. Therefore, the mass of the particles is inversely proportional to their acceleration due to gravity.

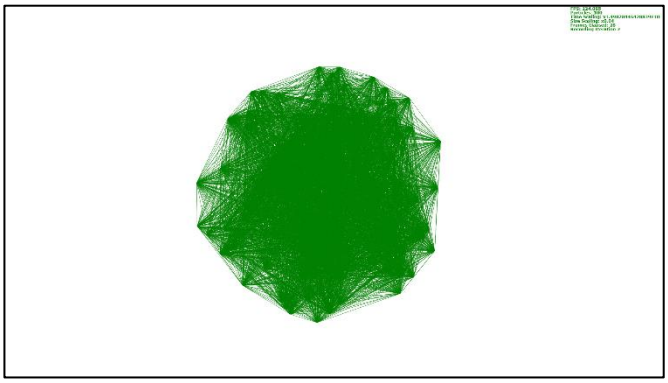
Nevertheless, the overall force acting on two particles with an increased mass still increases, if the fact that there are multiple particles is taken into consideration.

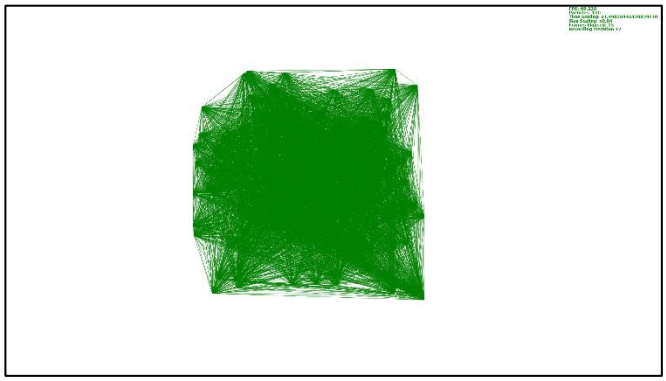
Hypothesis: *There is a positive correlation between the mass of the particles and the rate of coalescence.*

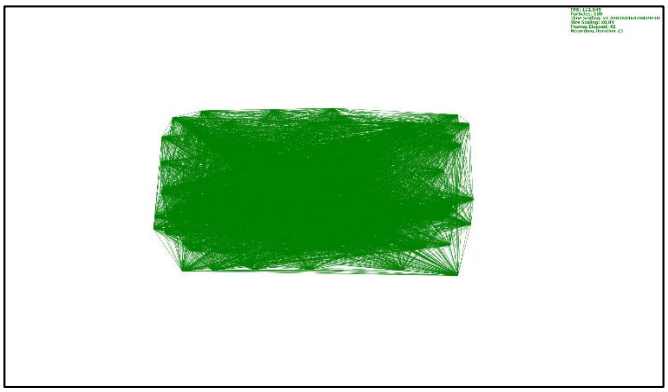
Conclusion: *There is a positive correlation between the mass of the particles and the rate of coalescence. (Hypothesis confirmed)*

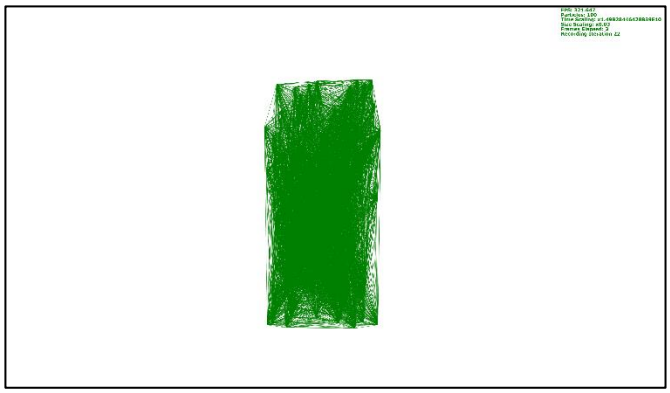
Influence of Distribution Shape on Coalescence Speed

Particle Number	Particle Mass (Kg)	Initial Speed (ms^{-1})	Particle Distribution	Frames Recorded
100	**100-1000**	**0**	**Circular, Square, Rectangular**	**50,000 (x30)**

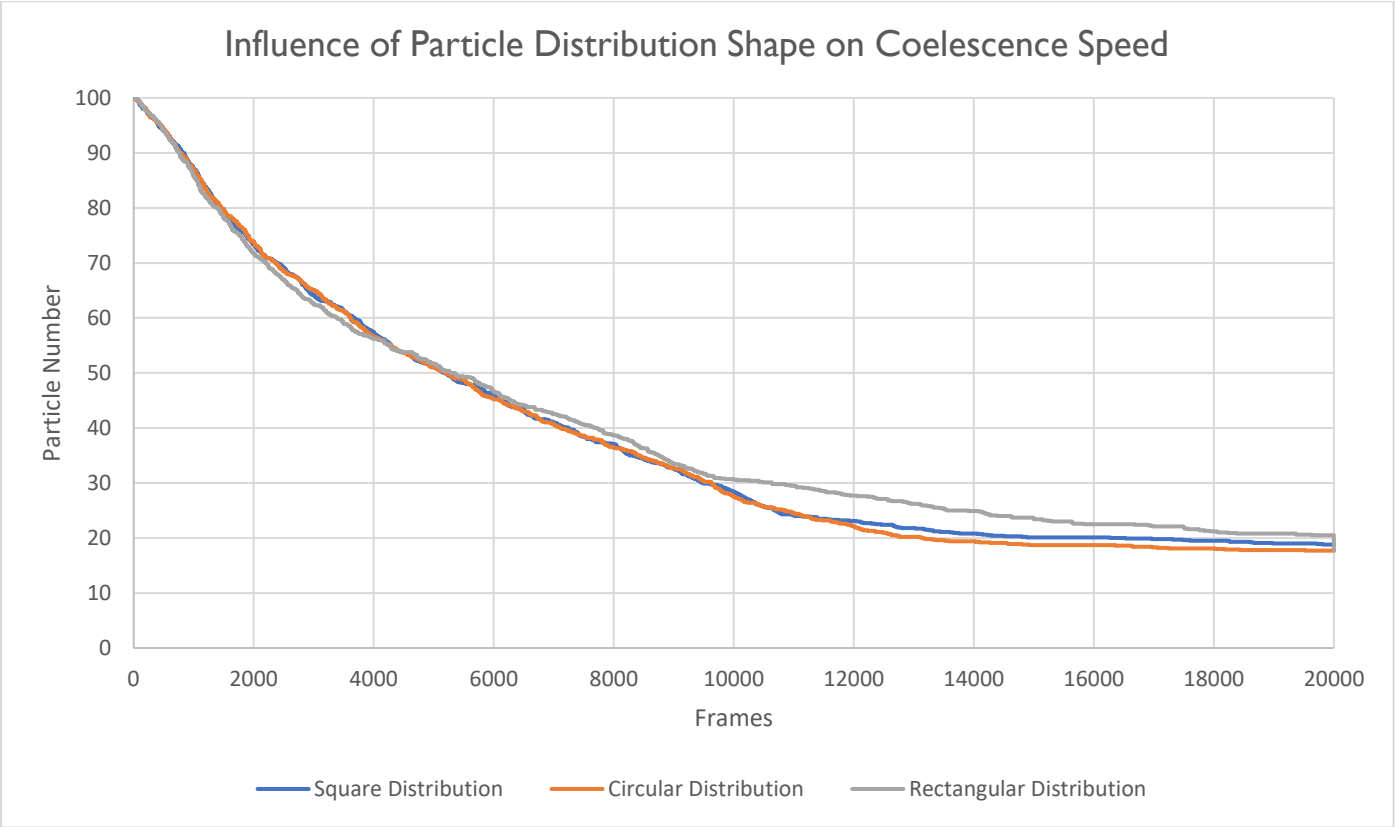








Alexey Elkin



The pictures above show the simulation running with different particle distribution shapes. In the first experiment, the particles are distributed in a circle. In the second, they are distributed in a square, and in the final experiment they are distributed in a rectangle (with sides 2:1). All of the distributions had the same area.

The purpose of the experiment is to identify the relationship between the shape of particle distribution and the rate at which the particles coalesce. The data for each of the 3 experiments was recorded 10 times, and the average was taken.

The graph shows that the particles in the circular distribution coalesced first, followed by the particles in the square and rectangular distributions.

This occurred because the particles in a circular distribution are evenly distributed around the centre of mass, whereas the particles in the square and rectangular distributions have some particles much further from the centre of mass than others. When these particles are pulled into the centre by gravity, they can "slingshot" past the centre and eject far off into space. This effect increases the time taken for all the particles to coalesce.

Hypothesis: *The particles in the circular distribution will coalesce the quickest, followed by the square, and then the rectangular distributions.*

Conclusion: *The particles in the circular distribution will coalesce the quickest, followed by the square, and then the rectangular distributions. However, the difference in the rate of coalescence does not differ by a significant amount when the area of the distribution remains the same.*

Overall Conclusion

After these experiments were performed, the following conclusions could be drawn:

- There is a positive correlation between the initial particle density and the rate of coalescence
- There is a negative correlation between the initial particle speed and the rate of coalescence
- There is a positive correlation between the mass of the particles and the rate of coalescence
- The more uniformly the particles are distributed around the centre, the faster the rate of coalescence.

Alexey Elkin

Selecting Lines of Best Fit

The lines of best fit were selected by assigning different regression models to the data (polynomial, exponential, quadratic, linear, etc…) and assessing the R^2 values. The line which produced the "best fit" (the lowest R^2 value) was chosen as the model or the data.

Experiment Limitations

Although the simulation used real physics laws, there were certain limitations that impeded the accuracy with which the simulation could be run. For example: the whole simulation was performed in 2D, and the FPS was dependent on thermal throttling caused by overheating. This meant that later experiments performed slightly slower than earlier ones because the computer I was using was purposefully lowering its processor frequency in order to protect components from overheating – slowing down the operation of the simulation.

There were also certain limitation to do with the processing power available. I used a **Dell XPS 15 7590** with **an Intel i7-9750H** processor and **32GB of RAM** to run all the experiments. However, only one processor core out of 6 was used to run the simulation, so full advantage of the processing power available was not taken. Theoretically, the application would have run much quicker on the GPU. However, this would have exacerbated the thermal issues – sacrificing reliability. Therefore, the limiting factor of the speed of operation of the simulation was the CPU temperature.

Safety and Risks

- PTSD from deleting half my program code by accident.

Code Access

The complete source code of the N-Body Simulation can be found on my GitHub page.

https://github.com/elkinal/Particles

https://github.com/elkinal/Particles_2.0 (Improved Version)

References

*Wikipedia Contributors (2020**a**). Newton's laws of motion. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Newton%27s_laws_of_motion*

*Wikipedia. (2020**b**). Newton. [online] Available at: https://en.wikipedia.org/wiki/Newton.*

*Wikipedia. (2020**c**). Inelastic collision. [online] Available at: https://en.wikipedia.org/wiki/Inelastic_collision [Accessed 10 May 2020].*