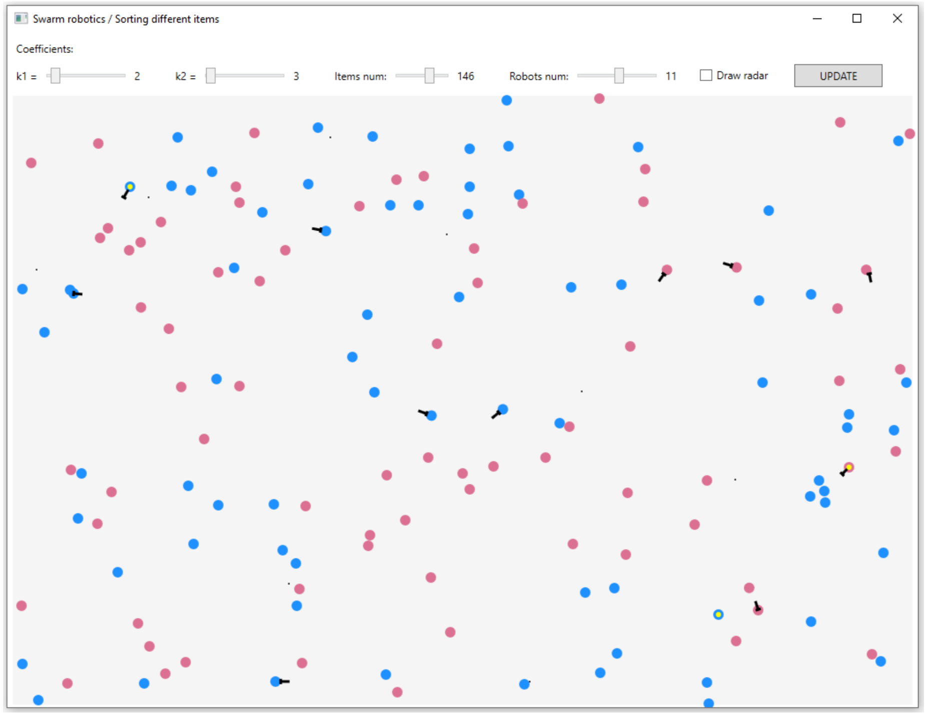
SWARM ROBOTICS

Sorting different objects


Swarm robotics / Sorting different items

Coefficients:

k1 = 2   k2 = 3   Items num: 146   Robots num: 11

Based on N.M.Ershov article "Swarm Robotics Algorithms"

SWARM ROBOTICS

Sorting different objects

Robots perform random movements in a given area. Each individual robot can perform the following two actions:

1. **Pick up** the found object;
2. **Drop** the item in the current location if it is free.

These actions are performed by robots in a probabilistic manner based on an analysis of their local neighborhood:

- the robot **picks up** an object it finds with a higher probability, the fewer objects of the same type it sees around it;

- the robot **drops** the object it is carrying with a probability that is greater, the more similar objects it sees around it.

SWARM ROBOTICS

Sorting different objects

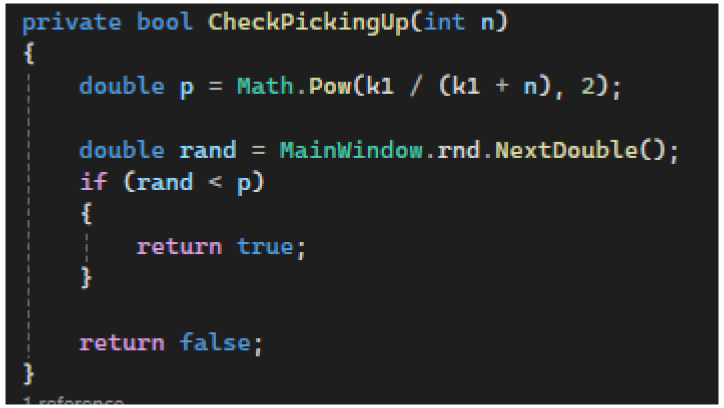
For the simplest case for objects of the same type, the probabilities of lifting **p** and dropping **q** are calculated based on the number **n** of objects located in the field of view of robot:

$$p = \left(\frac{k_1}{k_1 + n}\right)^2 \quad q = \left(\frac{n}{k_2 + n}\right)^2,$$

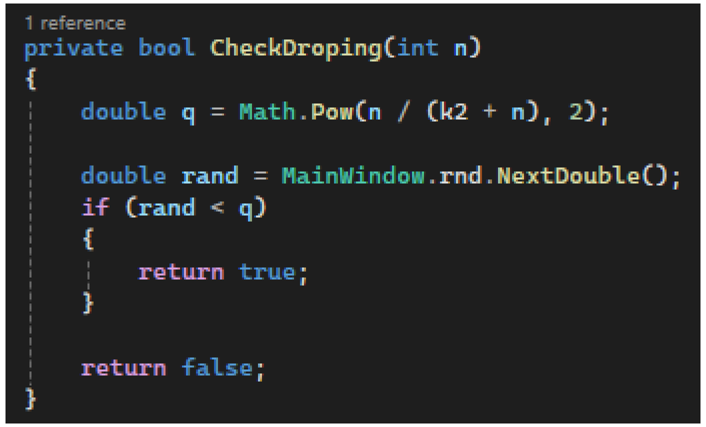
where **k1** > 0 and **k2** > 0 are the control parameters of the algorithm.

$$n = \max(0, n_0 - n_1),$$

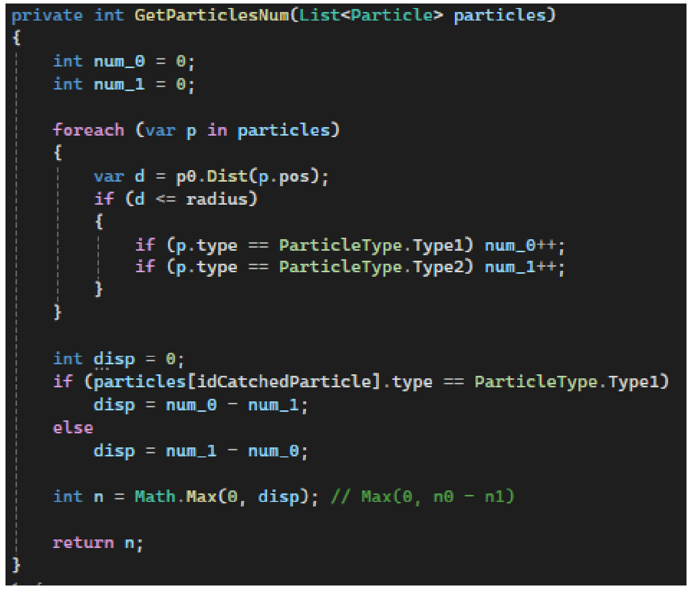
where **n0** is the number of items in the visibility area of the same type as the current item (which the robot wants to take or drop), and **n1** is the number of items of other types.



```
private bool CheckPickingUp(int n)
{
    double p = Math.Pow(k1 / (k1 + n), 2);

    double rand = MainWindow.rnd.NextDouble();
    if (rand < p)
    {
        return true;
    }

    return false;
}
```



```
1 reference
private bool CheckDroping(int n)
{
    double q = Math.Pow(n / (k2 + n), 2);

    double rand = MainWindow.rnd.NextDouble();
    if (rand < q)
    {
        return true;
    }

    return false;
}
```



```
private int GetParticlesNum(List<Particle> particles)
{
    int num_0 = 0;
    int num_1 = 0;

    foreach (var p in particles)
    {
        var d = p0.Dist(p.pos);
        if (d <= radius)
        {
            if (p.type == ParticleType.Type1) num_0++;
            if (p.type == ParticleType.Type2) num_1++;
        }
    }

    int disp = 0;
    if (particles[idCatchedParticle].type == ParticleType.Type1)
        disp = num_0 - num_1;
    else
        disp = num_1 - num_0;

    int n = Math.Max(0, disp); // Max(0, n0 - n1)

    return n;
}
```