GRAVITY

A MASSIVE PARTICLE SIMULATION
IMPLEMENTING QUADTREES

THE PROGRAM

This program is a solution to the ancient Greek N-body problem.
The N-Body Problem is a classical problem of predicting the individual motions of a group of celestial objects interacting with each other gravitationally.
Using this program, you can simulate the movement of thousands of particles interacting with each other gravitationally.

THE BUTTONS

Upon starting up Gravity^n, you will be faced with a whole load of buttons. Don't panic. Here you will find an explanation of all of them.

CLEAR

The clear button does what you would expect: It removes all particles from the system. All the settings, however, are retained.

RUN

The run button populates the system with particles. Once you have put particles in the system, you cannot change what dimension the simulation is in. You can press run while the simulation is paused, but the particles will not appear until the simulation is running.

PAUSE

The pause button halts the simulation. It does not change any particles' speed or position; it only pauses the physics.

PARTICLE COUNT

There are four buttons that control the number of particles you add to the system with each press of the run button. The x0.1, x0.5, x2, and x10 buttons flanking the “Particles to Add” indicator will change this number:

x0.1 divides it by 10
x0.5 halves it
x2 doubles it
x10 multiplies it by ten

This doesn’t allow for very exact numbers, but that is not the intent of this program: 10,000 particles simulates about the same as 10,001.
Note that there is a hard limit on the particle count, 200,000. Above that, Java may have some memory issues.

PARTICLE MASS

Like the particle count, there are four buttons that control the mass of particles in the system. The x0.1, x0.5, x2, and x10 buttons flanking the “Particle Mass” indicator will change this number, and behave exactly the same way the particle count buttons behave.
The mass buttons alter the mass of all the particles in the system, not just the ones to add next: Every particle in the program will always have the same mass as the next.

DIMENSION

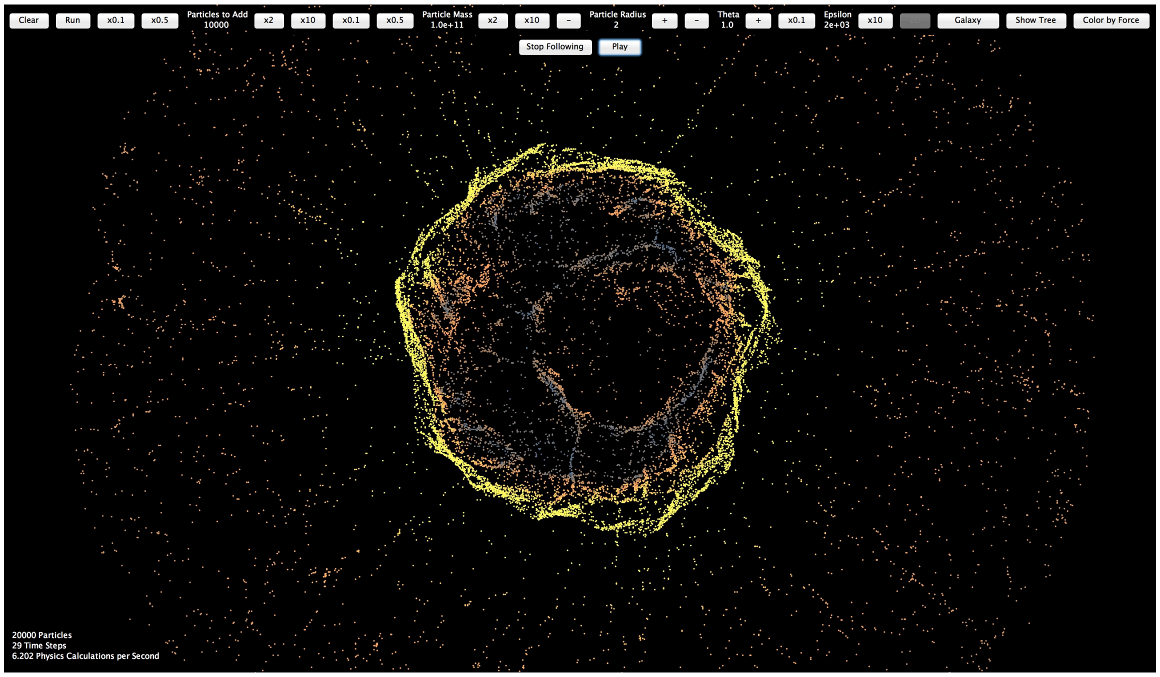
The button labeled 2D or 3D controls whether the particle system is simulated in two or three dimensions. Note that the button will be disabled if there are any particles already in the system, as jumping between dimensions is notoriously difficult.
The program will be a bit slower in three dimensions, so a higher theta value may be necessary for a smoother simulation.

FOLLOW CENTER

The Follow Center button determines whether the view is centered on the center of mass. If it is not toggled, the majority of particles may wander off screen; when it is toggled, a large proportion of the particles will usually be visible. Particles too far from the center of the screen (about five times the width of your screen) will be removed from the simulation.

COLOR BY FORCE

The Color By Force button will determine, predictably, whether the particles are colored by their net force. Dull blue particles have a very small net force acting on them, while orange and yellow particles have very large net forces.



```
20000 Particles
29 Time Steps
6.202 Physics Calculations per Second
```

A system of 20,000 particles after ~30 time steps, colored by net force

POPULATION RULES

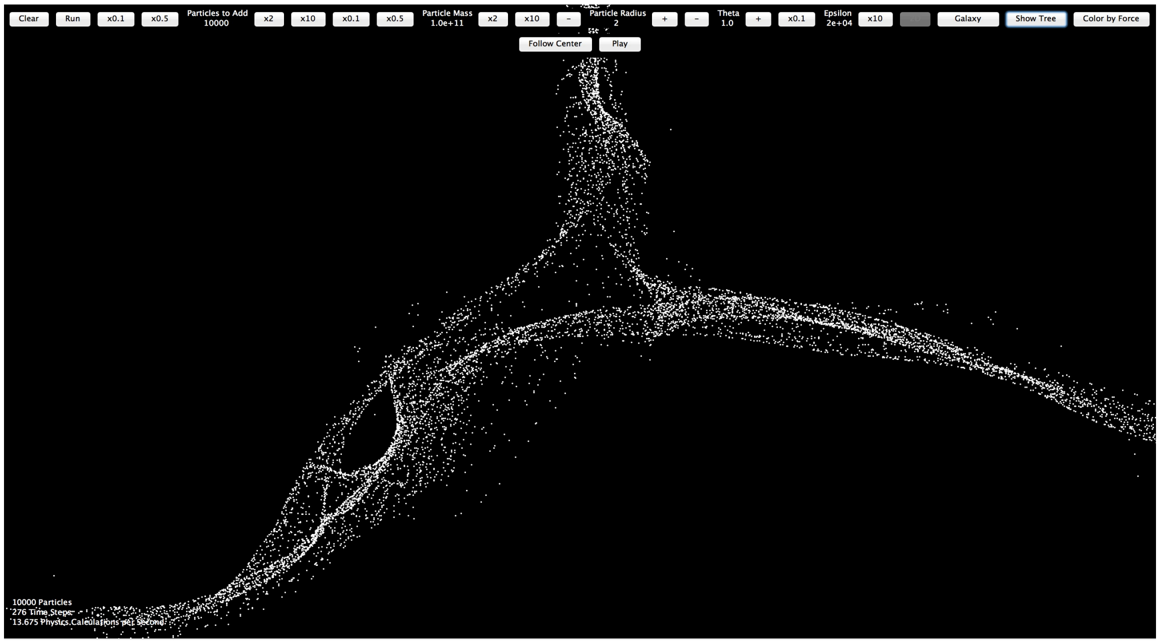
The button labeled either “Static Field” or “Galaxy” controls how particles will be put in the system. If the button shows “Static Field,” particles will be randomly placed throughout the window, with no initial velocity or force acting on them. If the button instead says “Galaxy,” the particles will be added within a circular disk, and given an initial velocity that will have them rotating the center of the disk.

PARTICLE RADIUS

The two buttons on either side of the “Particle Radius” indicator will decrement and increment the radius of the particles.
Note that the radius has no effect at all on the simulation itself, only the appearance of the particles.

SHOW / HIDE TREE

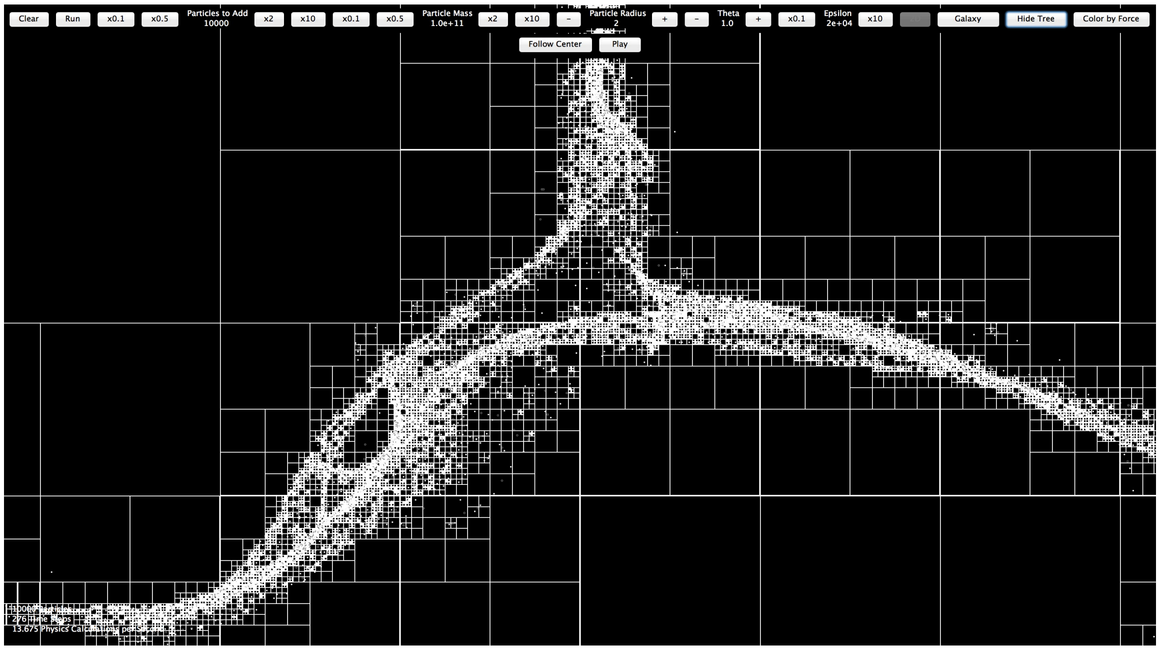
The show/hide tree button will determine whether the quadtree (or octree, depending on what dimension you're working in) will be displayed. Along with the bounds of each quadrant, particles are shown as solid white circles, and the center of mass of different quadrants as gray, open circles.



Particles to Add 10000
Particle Mass 1.0e+11
Particle Radius 2
Theta 1.0
Epsilon 2e+04

10000 Particles
276 Time Steps
13.675 Physics Calculations per Second

A system of 10,000 particles after ~300 time steps, quadtree hidden



Particles to Add 10000
Particle Mass 1.0e+11
Particle Radius 2
Theta 1.0
Epsilon 2e+04

10000 Particles
276 Time Steps
13.675 Physics Calculations per Second

The same system with the quadtree revealed

EPSILON

Epsilon is controlled the same way particle mass and count are. But what is epsilon?
Epsilon is the softening parameter of the simulation. In real-life physics, gravitational force is proportional to the product of the two masses acting on each other, divided by the distance between them squared. This means that, as particles approach each other, the force they exert on each other increases very quickly.
When computers simulate this force, they do a poor job: they accelerate the particles so quickly, they fly past each other without slowing back down. So, we need a softening parameter. The smaller it is, the more this computational error will crop up. The larger it is, the less accurate the simulation is.

THETA

Theta is controlled similarly to the particle radius, and it is the most important part of the program's implementation.
The simplest way to simulate a system of massive particles is brute forcing it: looking at each particle, and calculating the force acting on it from each other particle. This works fine with a few particles, but by the time you get into the thousands, it takes quite a long time to do all those calculations.
So, we use a quadtree. We divide the system of particles into four quadrants. Then, we subdivide each of those quadrants, and then we subdivide all of *those* quadrants, and continue subdividing until each quadrant has at most one particle. Then we go through each particle, the way we normally would, but only calculate the force from nearby particles. Distant particles are grouped together in their quadrants, and the force from their center of mass is instead calculated.
But what constitutes a particle being sufficiently far away to be summarized? Theta. If the ratio of the width of the quadrant being looked at to the distance between the particle and the quadrant's center of mass is less than theta, we summarize it.
So, if theta is set to zero, the simulation is essentially brute force, looking at each and every particle. If theta is very high, the simulation will run very smoothly, but extremely inaccurately. It is best to keep theta between zero and 1, balancing accuracy with speed.