```
1   using System;
2   using System.Collections;
3   using System.Collections.Generic;
4   using static Program.Constants;
5   using System.Threading;
6   using System.Threading.Tasks;
7   using System.Linq;
8   namespace Structures
9   {
10      public class PlanetarySystem : IEnumerable<Body> {
11                  protected bool running = false;
12                  protected List<Body> bodies;
13                  public List<int> centers {get; set;} = new List<int>();
14                  protected int center_index = -1; // -1 indicates space is not
    locked
15                  public PlanetarySystem(List<Body> bodies = null) {
16                          if (bodies == null) this.bodies = new List<Body>();
17                          else this.bodies = bodies;
18                  }
19                  public Body this[int key] {
20                  get {
21                          return this.bodies[key];
22                  }
23                  }
24                  public IEnumerator<Body> GetEnumerator() { return
    this.bodies.GetEnumerator(); }
25                  IEnumerator IEnumerable.GetEnumerator() { return
    this.bodies.GetEnumerator(); }
26                  public int Count {
27                          get {
28                                  return this.bodies.Count;
29                          }
30                  }
31                  public void Add(Body body) {
32                          bodies.Add(body);
33                  }
34                  public Vector3 Barycenter() {
35                          Vector3 weighted_center = Vector3.zero;
36                          double mu_total = 0;
37                          foreach (Body b in this) {
38                                  mu_total += b.stdGrav;
39                                  weighted_center += b.stdGrav*b.position;
40                          }
41                          return weighted_center/mu_total;
42                  }
43                  public void IterateCenter() {
44                          this.center_index += 1;
45              if (this.center_index >= this.centers.Count) {
46                  this.center_index = -1;
47              }
48                  }
49                  public Vector3 origin {
50                          get {
51                                  if (this.center_index == -1) return
    this.Barycenter();
52                                  else return this[this.centers
    [this.center_index]].position;
53                          }
54                  }
55                  protected Vector3[] GetAcceleration() {
56                          Vector3[] acceleration = new Vector3[this.Count];
57                          // Initialise our array to Vector3.zero, since the
    default is a null pointer.
58                          Parallel.For (0, this.Count, i => {
59                                  acceleration[i] = Vector3.zero;
60                          });
```

```
61                          for (int i = 0; i < this.Count; i++) {
62                                  // We will need the index later so foreach is
    not possible
63                                  Body body1 = this[i];
64                                  for (int j = i + 1; j < this.Count; j++) {
65                                          Body body2 = this[j]; // Again here
66                                          // The magnitude of the force,
    multiplied by G, = %mu_1 * %mu_2 / r^2
67                                          double mag_force_g = body1.stdGrav *
    body2.stdGrav / Math.Pow(Vector3.Magnitude(body1.position - body2.position),2);
68                                          // We lost direction in the previous
    calculation (since we had to square the vector), but we need it.
69                                          Vector3 direction = Vector3.Unit
    (body1.position - body2.position);
70                                          // since acceleration is F/m, and we
    have G*F and G*m, we can find an acceleration vector easily
71                                          Vector3 acceleration1 =  mag_force_g *
    -direction / body1.stdGrav;
72                                          Vector3 acceleration2 = mag_force_g *
    direction / body2.stdGrav;
73                                          acceleration[i] += acceleration1;
74                                          acceleration[j] += acceleration2;
75                                  }
76                          }
77                          return acceleration;
78                  }
79                  protected void TimeStep(double step) {
80                          var acceleration = this.GetAcceleration();
81                          for (int i = 0; i < acceleration.Length; i++) {
82                                  Body body = this[i];
83                                  Vector3 a = acceleration[i];
84                                  body.position += step*body.velocity + Math.Pow
    (step,2)*a/2;
85                                  body.velocity += step*a;
86                          }
87                  }
88                  public void StartAsync(double step = 1) {
89                          Task.Run(() => Start(step));
90                  }
91                  public void Start(double step = 1) {
92                          this.running = true;
93                          while (running) this.TimeStep(step);
94                  }
95                  public void Stop() {
96                          this.running = false;
97                  }
98          }
99  }
```