```
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                  // -1 indicates space is not locked
15                  public int center_index = -1;
16                  public PlanetarySystem(List<Body> bodies = null) {
17                          if (bodies == null) this.bodies = new List<Body>();
18                          else this.bodies = bodies;
19                  }
20
21
22                  public Body this[int key] {
23                  get {
24                          return this.bodies[key];
25                  }
26                  }
27                  public IEnumerator<Body> GetEnumerator() { return
    this.bodies.GetEnumerator(); }
28                  IEnumerator IEnumerable.GetEnumerator() { return
    this.bodies.GetEnumerator(); }
29                  public int Count {
30                          get {
31                                  return this.bodies.Count;
32                          }
33                  }
34                  public void Add(Body body) {
35                          bodies.Add(body);
36                  }
37                  public Vector3 Barycenter() {
38                          Vector3 weighted_center = Vector3.zero;
39                          double mu_total = 0;
40                          foreach (Body b in this) {
41                                  mu_total += b.stdGrav;
42                                  weighted_center += b.stdGrav*b.position;
43                          }
44                          return weighted_center/mu_total;
45                  }
46                  public void IterateCenter() {
47                          this.center_index += 1;
48              if (this.center_index >= this.centers.Count) {
49                  this.center_index = -1;
50              }
51                  }
52                  public Vector3 origin {
53                          get {
54                                  if (this.center_index == -1) return
    this.Barycenter();
55                                  else return this[this.centers
    [this.center_index]].position;
56                          }
57                  }
58                  protected Vector3[] GetAcceleration() {
59                          Vector3[] acceleration = new Vector3[this.Count];
60                          // Initialise our array to Vector3.zero, since the
    default is a null pointer.
61                          Parallel.For (0, this.Count, i => {
```

```
62                                      acceleration[i] = Vector3.zero;
63                              });
64                              for (int i = 0; i < this.Count; i++) {
65                                      // We will need the index later so foreach is
    not possible
66                                      Body body1 = this[i];
67                                      for (int j = i + 1; j < this.Count; j++) {
68                                              Body body2 = this[j]; // Again here
69                                              // The magnitude of the force,
    multiplied by G, = %mu_1 * %mu_2 / r^2
70                                              double mag_force_g = body1.stdGrav *
    body2.stdGrav / Math.Pow(Vector3.Magnitude(body1.position -
    body2.position),2);
71                                              // We lost direction in the previous
    calculation (since we had to square the vector), but we need it.
72                                              Vector3 direction = Vector3.Unit
    (body1.position - body2.position);
73                                              // since acceleration is F/m, and we
    have G*F and G*m, we can find an acceleration vector easily
74                                              Vector3 acceleration1 =  mag_force_g
    * -direction / body1.stdGrav;
75                                              Vector3 acceleration2 = mag_force_g *
    direction / body2.stdGrav;
76                                              acceleration[i] += acceleration1;
77                                              acceleration[j] += acceleration2;
78                                      }
79                              }
80                              return acceleration;
81                      }
82                      protected void TimeStep(double step) {
83                              var acceleration = this.GetAcceleration();
84                              for (int i = 0; i < acceleration.Length; i++) {
85                                      Body body = this[i];
86                                      Vector3 a = acceleration[i];
87                                      body.position += step*body.velocity + Math.Pow
    (step,2)*a/2;
88                                      body.velocity += step*a;
89                              }
90                      }
91                      public void StartAsync(double step = 1) {
92                              Task.Run(() => Start(step));
93                      }
94                      public void Start(double step = 1) {
95                              this.running = true;
96                              while (running) this.TimeStep(step);
97                      }
98                      public void Stop() {
99                              this.running = false;
100                     }
101             }
102     }
```