```
1   using System;
2   using System.Collections.Generic;
3   using static Program.Constants;
4   using System.Threading;
5   using System.Threading.Tasks;
6   using System.Linq;
7   namespace Structures
8   {
9           [Serializable()]
10          public class Body : ICloneable {
11                  public string name {get; set;}
12                  public Body parent {get; set;}
13                  public double stdGrav {get; set;} // standard gravitational
    parameter
14                  public double radius {get; set;}
15                  public Vector3 position {get; set;} = Vector3.zero;
16                  public Vector3 velocity {get; set;} = Vector3.zero;
17                  public Vector3 color {get; set;} = new Vector3(1,1,1);
18                  public Body() {} // paramaterless constructor for
    serialisation
19                  public Body (Body parent, OrbitalElements elements) {
20                          // First check the values are reasonable. If parent
    == null it is assumed that
21                          // position and velocity are set explicitly, and this
    constructor is not used
22                          if (parent == null) return;
23                          this.parent = parent;
24                          if (elements.eccentricity < 0
25                           || elements.semilatusrectum < 0
26                           || elements.inclination < 0
27                           || elements.inclination > Math.PI
28                           || elements.ascendingNodeLongitude < 0
29                           || elements.ascendingNodeLongitude >= 2*Math.PI
30                           || elements.periapsisArgument < 0
31                           || elements.periapsisArgument >= 2*Math.PI
32                           || elements.trueAnomaly < 0
33                           || elements.trueAnomaly >= 2*Math.PI
34                          ){
35                                  // Throw an exception if the arguments are
    out of bounds
36                                  throw new ArgumentException();
37                          }
38                          // working in perifocal coordinates (periapsis along
    the x axis, orbit in the x,y plane):
39                          double mag_peri_radius = elements.semilatusrectum/(1
    +elements.eccentricity*Math.Cos(elements.trueAnomaly));
40                          Vector3 peri_radius = mag_peri_radius*new Vector3
    (Math.Cos(elements.trueAnomaly),Math.Sin(elements.trueAnomaly),0);
41                          Vector3 peri_velocity = Math.Sqrt(parent.stdGrav/
    elements.semilatusrectum)
42                                                                          * new
    Vector3(
43                                                                          -
    Math.Sin(elements.trueAnomaly),
44
    Math.Cos(elements.trueAnomaly) + elements.eccentricity,
45
    0
46                                                                          );
47                          // useful constants to setup transformation matrix
48                          var sini = Math.Sin(elements.inclination); // i <-
    inclination
49                          var cosi = Math.Cos(elements.inclination);
50                          var sino = Math.Sin
    (elements.ascendingNodeLongitude); // capital omega <- longitude of ascending
    node
```

```
51                          var coso = Math.Cos(elements.ascendingNodeLongitude);
52                          var sinw = Math.Sin(elements.periapsisArgument); //
    omega <- argument of periapsis
53                          var cosw = Math.Cos(elements.periapsisArgument);
54                          // Transform perifocal coordinates to i,j,k
    coordinates
55                          Matrix3 transform = new Matrix3(
56                                  new Vector3(
57                                          coso*cosw - sino*sinw*cosi,
58                                          -coso*sinw-sino*cosw*cosi,
59                                          sino*sini
60                                  ),
61                                  new Vector3(
62                                          sino*cosw+coso*sinw*cosi,
63                                          -sino*sinw+coso*cosw*cosi,
64                                          -coso*sini
65                                  ),
66                                  new Vector3(
67                                          sinw*sini,
68                                          cosw*sini,
69                                          cosi
70                                  )
71                          );
72                          // add the parent's position and velocity since that
    could be orbiting something too
73                          this.position = transform*peri_radius +
    parent.position;
74                          this.velocity = transform*peri_velocity +
    parent.velocity;
75                  }
76                  public double HillRadius() {
77                          // This is the maximum distance anything can
    reasonably orbit at.
78                          // It would normally depend on the bodies nearby, but
    we'll just do something simple
79                          // which is roughly accurate for bodies in the solar
    system.
80                          return this.stdGrav * 1e-6;
81                  }
82                  public object Clone() {
83                          return new Body {
84                                  name = this.name,
85                                  parent = this.parent,
86                                  stdGrav = this.stdGrav,
87                                  radius = this.radius,
88                                  position = this.position,
89                                  velocity = this.velocity,
90                                  color = this.color
91                          };
92                  }
93          }
```