```
(*sphere/sphere collision resolver*)
p1 = {p1.x, p1.y, p1.z};(*position*)
u1 = {u1.x, u1.y, u1.z};(*velocity*)
r1 = r1; (*radius*)
p2 = {p2.x, p2.y, p2.z};
r2;
u2 = {u2.x, u2.y, u2.z};
Solve[EuclideanDistance[p1 + u1 t, p2 + u2 t] == r1 + r2, t];
(*if overlapping pick nearest t≤0, for collision detection check t>0 and t<1*)
np1 = p1 + u1 t;(*move out of collision*)
np2 = p2 + u2 t;
nml = Normalize[np1 - np2];(*collision plane normal*)
m1;(*mass*)
m2;
(*     velocity given    velocities received along collision plane normal*)
v1 = u1 - (u1.nml) nml + (u1.nml) nml ((m1 - m2) / (m1 + m2)) + (u2.nml) nml (2 m2 / (m1 + m2));
v2 = u2 - (u2.nml) nml + (u2.nml) nml ((m2 - m1) / (m1 + m2)) + (u1.nml) nml (2 m1 / (m1 + m2));
np1 = np1 + v1 (1 - t);(*perform remaining dt with new velocities*)
np2 = np2 + v2 (1 - t);

m1 u1 + m2 u2 == m1 v1 + m2 v2;
```