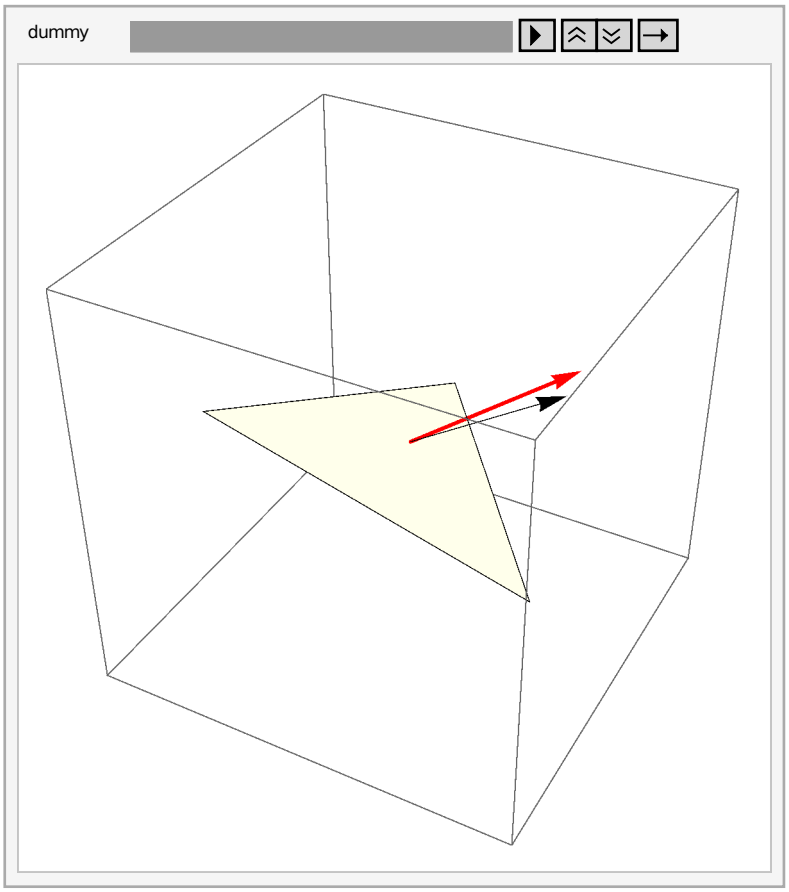
Initially, the prinicpal axes e1, e2, e3 coicide with the x,y,z axes. The rigid triangle is represented as three vertex positions, ra, rc, rd, in the x,y,z coordinate system, which is centered on the triangle's CoM. After each time step of size dt, we update each position vector with r := r + dt * velocity, where velocity = $\omega \times$ r, and then we update ω according to the Euler equations. We repeat this many times to generate a list of images, forming an animation. In order to move the ω vector from body frame to space frame, I keep track of the "direction cosines" that map each body-frame unit vector into the space frame. In other words, the vector e1 starts out as {1,0,0} but at any given time contains the space-frame components of the e1 unit vector, and so on for e2 and e3.

In[1]:=
```
(* The body unit vectors, e1, e2, e3, are
    initiallyalong the x,y,z space axes. *)
e1 = {1, 0, 0};
e2 = {0, 1, 0};
e3 = {0, 0, 1};
(* Time  step for each frame  of the animation.
     We want to keep the product ω dt small ,
   since we're using the expressions for
    infinitesimal rotations. *)
dt = 0.01;
(* The initial(space frame ) positions of the
  three vertices, A, C, D, of the triangle. *)
ra = {-2, -1, 0};
rc = {+2, -1, 0};
rd = {0, +1, 0};
(* Principal moments   of inertia *)
λ1 = 6; λ2 = 8; λ3 = 14;
(* Function to rotate vector x by infinitesimal
    rotation vector ωdt *)
rot[x_, ωdt_] := Block[{norm },
      norm  = Norm [x];
      xrot = x + ωdt * x;
      xrot * norm  / Norm [xrot]
    ];
(* Initial value of rotation vector ω in the
    body frame , which initiallycoincides with
    the space frame . *)
ω_body = {2.0, 3.0, 0.0};
ω_body = ω_body / Norm [ω_body];
(* Save the initialmagnitude  of angular-momentum
    vector.  I do this so that I can scale the displayed
    vector to a size that is easy to see, in such a way
   that we would still notice if some  bug caused the
   magnitude  of the angular momentum    to change.  The
   angular momentum    is shown as a red arrow. *)
l_body0 = ω_body * {λ1, λ2, λ3};
l_space0 = l_body0; (* Since frames  initiallycoincide *)
l_magnitude0  = Norm [l_body0];
(* Save the initialmagnitude  of the rotation vector,
   so that we can display a scaled version of this
    vector as a (black) arrow on the animation . *)
ω_magnitude0  = Norm [ω_body];
(* Shortcut for origin of coordinate system . *)
o = {0, 0, 0};
(* This "function" simply  updates the pertinent variables,
```

```
    according to the Euler equations, for each time  step.
      Its return "value" is the next frame  of the animation ,
    i.e. a 3D graph showing the current orientation of the
     triangle, with the angular-momentum    and ω vectors drawn
      as arrows.  All of this is drawn in the space frame . *)
nexttriangle := Block[{},
       (* Project rotation vector into space frame . *)
       ωspace = ωbody[[1]] e1 + ωbody[[2]] e2 + ωbody[[3]] e3;
       ωdt = ωspace dt;
       (* Update the space-frame  triangle vertex positions *)
       ra = rot[ra, ωdt];
       rc = rot[rc, ωdt];
       rd = rot[rd, ωdt];
       (* Update the space-frame  representations of the
            three body-axis unit vectors. *)
       e1 = rot[e1, ωdt];
       e2 = rot[e2, ωdt];
       e3 = rot[e3, ωdt];
       (* Use the Euler equations of motion  (torque-free)
          to update the rotational velocity, as evaluated
          in the body frame . *)
       ω1dot = ωbody[[2]] * ωbody[[3]] * (λ2 - λ3)/λ1;
       ω2dot = ωbody[[3]] * ωbody[[1]] * (λ3 - λ1)/λ2;
       ω3dot = ωbody[[1]] * ωbody[[2]] * (λ1 - λ2)/λ3;
       ωbody = ωbody + {ω1dot, ω2dot, ω3dot}dt;
       (* Recalcualte angular-momentum    vector in body frame ,
          then project it to the space frame , as a check to
           make  sure simulation  is working.  The lspace
           vector should stay constant if all is well. *)
       lbody = {ωbody[[1]] λ1, ωbody[[2]] λ2, ωbody[[3]] λ3};
       lspace = lbody[[1]] e1 + lbody[[2]] e2 + lbody[[3]] e3;
       (* Draw (in space frame ) the triangle vertices,
          the angular-velocity vector (black), and the
            angular-momentum    vector (red).  This graphic
            constitutes one frame  of the animation . *)
       Graphics3D[
         {Triangle[{ra, rc, rd}],
           Arrow[{o, 2*ωspace/ωmagnitude0 }],
           Thick, Red, Arrow[{o, 2.4*lspace/lmagnitude0 }]},
         PlotRange → {{-2.5, 2.5}, {-2.5, 2.5}, {-2.5, 2.5}}]
    ];
(* I have no idea why I needed to create a Block here
  with an evaluation of "dummy  " inside of it.  It didn't
  seem  to work when I had only "nexttriangle" inside
  the Animate [] call. *)
Animate [Block[{},
    dummy  ;
    nexttriangle], {dummy  , 0, Infinity}]
```

Out[18]=




```
(* I did this to export the movie  files *)
(*
frames  = Table[nexttriangle, {dummy  , 0, 10000}];
Export["strucktriangle.avi", frames [[Range[1,10000, 50]]]];
ListAnimate[frames , 20]
*)
```