Run the entire notebook once. Then for each moment tensor, you need only run the last command, after entering your own eigenvalue triple Λ and orthonormal eigenbasis u.

See the file BeachballCodeTutorial for more explanation.

```
sind[t_] := Sin[t Degree];
cosd[t_] := Cos[t Degree];
tand[t_] := Tan[t Degree];
arccosd[t_] := ArcCos[t] / Degree;
arctand[t_] := ArcTan[t] / Degree;
xyztp[{θ_, ϕ_}] := {cosd[θ] sind[ϕ], sind[θ] sind[ϕ], cosd[ϕ]};
id = IdentityMatrix[3];
BasicPoly[i_, j_, RangeX_, RangeY_] := {{RangeX[[i   ]], RangeY[[j]]},
                                        {RangeX[[i + 1]], RangeY[[j]]},
                                        {RangeX[[i + 1]], RangeY[[j + 1]]},
                                        {RangeX[[i   ]], RangeY[[j + 1]]}};
BasicPolys[RangeX_, RangeY_] := Flatten[Table[BasicPoly[i, j, RangeX, RangeY],
    {i, Length[RangeX] - 1}, {j, Length[RangeY] - 1}], 1];
```

$$\texttt{XRot[t\_] :=} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \texttt{Cos[t]} & -\texttt{Sin[t]} \\ 0 & \texttt{Sin[t]} & \texttt{Cos[t]} \end{pmatrix};$$

$$\texttt{YRot[t\_] :=} \begin{pmatrix} \texttt{Cos[t]} & 0 & \texttt{Sin[t]} \\ 0 & 1 & 0 \\ -\texttt{Sin[t]} & 0 & \texttt{Cos[t]} \end{pmatrix};$$

$$\texttt{ZRot[t\_] :=} \begin{pmatrix} \texttt{Cos[t]} & -\texttt{Sin[t]} & 0 \\ \texttt{Sin[t]} & \texttt{Cos[t]} & 0 \\ 0 & 0 & 1 \end{pmatrix};$$

```
xrot[t_] := XRot[t Degree];
yrot[t_] := YRot[t Degree];
zrot[t_] := ZRot[t Degree];
```

$$\texttt{rot111 =} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix};$$

```
zref = DiagonalMatrix[{1, 1, -1}];
myRange[x1_, x2_, Nsubintervals_] :=
  If[x2 <= x1, {}, Range[x1, x2, (x2 - x1) / Nsubintervals]];
HueIn = White;  (* change to GrayLevel[.8] or Blue, etc. *)
HueOut = Red;
HueBeach[v_, M_] := If[(M.v).v > 0, HueOut, HueIn];
```

$$\texttt{phiN[}\theta\texttt{\_, \{}\lambda\texttt{1\_, }\lambda\texttt{2\_, }\lambda\texttt{3\_\}] := arccosd}\left[\sqrt{\frac{\lambda1\,\texttt{cosd}[\theta]^2 + \lambda2\,\texttt{sind}[\theta]^2}{\lambda1\,\texttt{cosd}[\theta]^2 + \lambda2\,\texttt{sind}[\theta]^2 - \lambda3}}\right]$$

```
SubdivideForLambda[poly_, {λ1_, λ2_, λ3_}] :=
  If[(λ1 ≥ 0 && λ2 ≥ 0 && λ3 ≥ 0) || (λ1 <= 0 && λ2 <= 0 && λ3 <= 0),
   poly, Subdivide[poly, phiN[#, {λ1, λ2, λ3}] &]];
Subdivide[{{x1_, y1_}, {x2_, y2_}, {x3_, y3_}, {x4_, y4_}}, f_] :=
  If[y1 < f[x1] < y4 && y2 < f[x2] < y3,
```

```
{{{x1, y1}, {x2, y2}, {x2, f[x2]}, {x1, f[x1]}},     (* lower quadrilateral *)
 {{x4, y4}, {x3, y3}, {x2, f[x2]}, {x1, f[x1]}}},   (* upper quadrilateral *)

If[y1 < f[x1] < y4 && f[x2] < y2,
 {{{x1, f[x1]}, {x1, y1}, {x1 + (y1 - f[x1]) (x2 - x1)/(f[x2] - f[x1]), y1}, {x1, f[x1]}},
   (* upper left triangle *)
   {{x1, f[x1]}, {x1 + (y1 - f[x1]) (x2 - x1)/(f[x2] - f[x1]), y1}, {x2, y2}, {x3, y3}, {x4, y4}}},
 (* the remaining pentagon *)

 If[f[x1] < y1 && y2 < f[x2] < y3,
  {{{x1 + (y1 - f[x1]) (x2 - x1)/(f[x2] - f[x1]), y1}, {x2, y2}, {x2, f[x2]}},
    (* upper right triangle *)
    {{x1 + (y1 - f[x1]) (x2 - x1)/(f[x2] - f[x1]), y1}, {x2, f[x2]}, {x3, y3}, {x4, y4}, {x1, y1}}},

  If[y4 < f[x1] && y2 < f[x2] < y3,
   {{{x1 + (y3 - f[x1]) (x2 - x1)/(f[x2] - f[x1]), y3}, {x3, y3}, {x3, f[x3]}},
     (* lower right triangle *)
     {{x1 + (y3 - f[x1]) (x2 - x1)/(f[x2] - f[x1]), y3}, {x3, f[x3]}, {x2, y2}, {x1, y1}, {x4, y4}}},

   If[y1 < f[x1] < y4 && y3 < f[x2],
    {{{x1 + (y3 - f[x1]) (x2 - x1)/(f[x2] - f[x1]), y3}, {x4, y4}, {x4, f[x4]}},
      (* lower left triangle *)
      {{x1 + (y3 - f[x1]) (x2 - x1)/(f[x2] - f[x1]), y3}, {x3, y3}, {x2, y2},
       {x1, y1}, {x1, f[x1]}}},

    If[(f[x2] < y2 && y4 < f[x4]) || (f[x1] < y1 && y3 < f[x3]),
     {{{x4 + (y1 - f[x4]) (x2 - x4)/(f[x2] - f[x4]), y1}, {x2, y2}, {x3, y3},
        {x4 + (y4 - f[x4]) (x2 - x4)/(f[x2] - f[x4]), y3}}, (* left quadrilateral *)
       {{x4 + (y1 - f[x4]) (x2 - x4)/(f[x2] - f[x4]), y1}, {x1, y1}, {x4, y4},
        {x4 + (y4 - f[x4]) (x2 - x4)/(f[x2] - f[x4]), y3}}},
```

```
        {{{x1, y1}, {x2, y2}, {x3, y3}, {x4, y4}}}]]]]]];

th0[{a_, b_, c_}] := arctand[Sqrt[-a / b]]; (* special case for λ1 λ2 λ3 = 0 *)
OrangeSlice[θ1_, θ2_] :=
  Map[xyztp[#] &, BasicPolys[myRange[θ1, θ2, Min[Select[Range[30], (θ2 - θ1)/# ≤ 10 &]]],
    Range[0., 180, 10]], {2}];

PolyList = BasicPolys[Range[0., 360, 10], Range[0., 90, 10]];
(* (θ, ϕ) polys for hemisphere; take smaller increments if desired *)
SubdividePolyList[PolyList_, Λ_] :=
  Flatten[SubdivideForLambda[#, Λ] & /@ PolyList, 1];
(* (θ, ϕ) polys for hemisphere after the subdivision *)

PolyListForBeachball0[Λ_] := If[(Λ[[1]] ≥ 0 && Λ[[2]] ≥ 0 && Λ[[3]] ≥ 0) ||
     (Λ[[1]] <= 0 && Λ[[2]] <= 0 && Λ[[3]] <= 0), (* all white or all red *)
    Join[Map[zref.xyztp[#] &, PolyList, {2}],
        Map[     xyztp,      PolyList, {2}]],
    If[Λ[[1]] Λ[[2]] Λ[[3]] == 0, (* at least one eigenvalue is zero *)
     Join[OrangeSlice[-th0[Λ], th0[Λ]],
      OrangeSlice[th0[Λ], -th0[Λ] + 180], OrangeSlice[-th0[Λ] + 180,  th0[Λ] + 180],
      OrangeSlice[ th0[Λ] + 180, -th0[Λ] + 360]],
     Join[                          (* the usual
       case *)
      Map[zref.xyztp[#] &, SubdividePolyList[PolyList, Λ], {2}],
      Map[     xyztp,      SubdividePolyList[PolyList, Λ], {2}]]]];
```

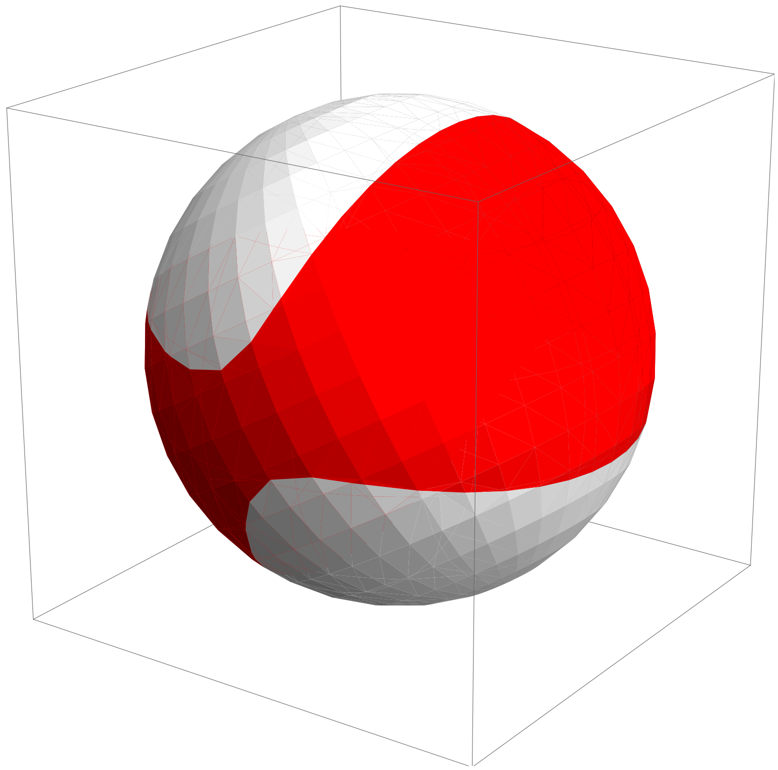
PolyListForBeachball0 was only for $\lambda 1 >= 0, \lambda 2 >= 0, \lambda 3 <= 0$ OR $\lambda 1 <= 0, \lambda 2 <= 0, \lambda 3 >= 0$ (or for all $\lambda$s the same sign). PolyList-ForBeachball removes the restriction.

```
CycleMat[{a_, b_, c_}] :=
  If[(a > 0 && c > 0 && b < 0) || (a < 0 && c < 0 && b > 0) || b == 0, rot111,
   If[(b > 0 && c > 0 && a < 0) || (b < 0 && c < 0 && a > 0) || a == 0, rot111.rot111, id]];
PolyListForBeachball[Λ_] :=
  Map[Transpose[CycleMat[Λ]].# &, PolyListForBeachball0[CycleMat[Λ].Λ], {2}];
```

Beachball[Λ, u, c, BBrad] is for the moment tensor $[\Lambda]_U$. BBrad and c are the radius and center of the ball.

```
BeachBall[Λ_, u_, c_, BBrad_] :=
  {HueBeach[Mean[(# - c) & /@ #], u.DiagonalMatrix[Λ].Transpose[u]], EdgeForm[],
     Polygon[#]} & /@ Map[c + BBrad * u.# &, PolyListForBeachball[Λ], {2}];
```

```
Λ = {1, 19, -20}; u = xrot[30]; eye = 5 xyztp[{35, 70}]; c = {10, 0, 0}; BBrad = 1;
Graphics3D[BeachBall[Λ, u, c, BBrad], ViewPoint → eye, Lighting → "Neutral"]
```



The beachball can be rotated with the mouse.

You can experiment with other choices of Λ and u.