Triple Pendulum Simulation

Initialization Code (optional)

Manipulate

```
Manipulate[
 Row[{Dynamic[Refresh[Which[state == "RESET",
      (
       currentTime = 0;
       tick = 0;

       {currentPE, currentKE, phasePortraitPlot, bob1, bob2, bob3, θ1, θ1Speed} =
        update[nBobs, θ1Init, θ2Init, θ3Init, θ1SpeedInit, θ2SpeedInit, θ3SpeedInit,
         m1, m2, m3, L1, L2, L3, g, currentTime, c, maxRunTime, dt, showPhase];

       state = "PAUSE"
      ),

      state == "PAUSE",
      (
       Which[lastEvent == "duration_changed" || lastEvent == "delt_changed",
         (
          lastEvent = "no_event";

          If[showPhase,
           phasePortraitPlot = makePhasePortrait[nBobs, θ1Init, θ2Init, θ3Init,
             θ1SpeedInit, θ2SpeedInit, θ3SpeedInit, m1, m2, m3, L1, L2, L3, g, c, maxRunTime, dt]
          ]
         ),

        lastEvent == "show_phase",
         (
          lastEvent = "no_event";
          phasePortraitPlot = makePhasePortrait[nBobs, θ1Init, θ2Init, θ3Init,
            θ1SpeedInit, θ2SpeedInit, θ3SpeedInit, m1, m2, m3, L1, L2, L3, g, c, maxRunTime, dt]
         ),

        lastEvent == "run_button", (lastEvent = "no_event"; state = "RUNNING"; tick += DEL),

        lastEvent == "initial_conditions_changed",
         (
          lastEvent = "no_event";
          currentTime = 0;

          {currentPE, currentKE, phasePortraitPlot, bob1, bob2, bob3, θ1, θ1Speed} =
           update[nBobs, θ1Init, θ2Init, θ3Init, θ1SpeedInit, θ2SpeedInit, θ3SpeedInit,
            m1, m2, m3, L1, L2, L3, g, currentTime, c, maxRunTime, dt, showPhase]
         ),

        lastEvent == "step_button",
         (
          lastEvent = "no_event";
```

```
   {currentPE, currentKE, phasePortraitPlot, bob1, bob2, bob3, θ1, θ1Speed} =
    update[nBobs, θ1Init, θ2Init, θ3Init, θ1SpeedInit, θ2SpeedInit, θ3SpeedInit,
     m1, m2, m3, L1, L2, L3, g, currentTime, c, maxRunTime, dt, showPhase];

   If[currentTime + dt > maxRunTime,
    (
     state = "RESET";
     tick += DEL
    ),
    (
     currentTime += dt
    )
   ]
  ),
  lastEvent == "reset_button", (lastEvent = "no_event"; state = "RESET"; tick += DEL),
  lastEvent == "pause_button", lastEvent = "no_event",
  lastEvent == "mouseDown", lastEvent = "no_event",
  lastEvent == "mouseUp",
  (
   lastEvent = "no_event";
   currentTime = 0
  )
 ]
),

state == "RUNNING",
(
 Which[lastEvent == "show_phase",
  (
   lastEvent = "no_event";
   phasePortraitPlot = makePhasePortrait[nBobs, θ1Init, θ2Init, θ3Init,
     θ1SpeedInit, θ2SpeedInit, θ3SpeedInit, m1, m2, m3, L1, L2, L3, g, c, maxRunTime, dt];
   tick += DEL
  ),

  lastEvent == "duration_changed" || lastEvent == "delt_changed",
  (
   lastEvent = "no_event";
   If[showPhase,
    phasePortraitPlot = makePhasePortrait[nBobs, θ1Init, θ2Init, θ3Init,
      θ1SpeedInit, θ2SpeedInit, θ3SpeedInit, m1, m2, m3, L1, L2, L3, g, c, maxRunTime, dt]
   ];
   tick += DEL
  ),

  lastEvent == "no_event" || lastEvent == "run_button",
  (
   If[lastEvent == "run_button", lastEvent = "no_event"];

   {currentPE, currentKE, phasePortraitPlot, bob1, bob2, bob3, θ1, θ1Speed} =
    update[nBobs, θ1Init, θ2Init, θ3Init, θ1SpeedInit, θ2SpeedInit, θ3SpeedInit,
     m1, m2, m3, L1, L2, L3, g, currentTime, c, maxRunTime, dt, showPhase];

   If[currentTime + dt > maxRunTime,
    (
     currentTime = 0
    ),
    (
     currentTime += dt
    )
```

```
           ];

           tick += DEL
          ),

          lastEvent == "pause_button",
          (
           lastEvent = "no_event";
           state = "PAUSE"
          ),

          lastEvent == "mouseDown", (lastEvent = "no_event"),

          lastEvent == "mouseUp",
          (
           lastEvent = "no_event";
           currentTime = 0;
           tick += DEL
          ),

          lastEvent == "reset_button", (lastEvent = "no_event"; state = "RESET"; tick += DEL),

          lastEvent == "step_button",
          (
           lastEvent = "no_event";

           {currentPE, currentKE, phasePortraitPlot, bob1, bob2, bob3, θ1, θ1Speed} =
            update[nBobs, θ1Init, θ2Init, θ3Init, θ1SpeedInit, θ2SpeedInit, θ3SpeedInit,
             m1, m2, m3, L1, L2, L3, g, currentTime, c, maxRunTime, dt, showPhase];

           If[currentTime + dt > maxRunTime, currentTime = 0, currentTime += dt];
           state = "PAUSE"
          ),

          lastEvent == "initial_conditions_changed",
          (
           lastEvent = "no_event";
           currentTime = 0;
           {currentPE, currentKE, phasePortraitPlot, bob1, bob2, bob3, θ1, θ1Speed} =
            update[nBobs, θ1Init, θ2Init, θ3Init, θ1SpeedInit, θ2SpeedInit, θ3SpeedInit,
             m1, m2, m3, L1, L2, L3, g, currentTime, c, maxRunTime, dt, showPhase];

           tick += DEL
          ),

          lastEvent == "pause_button", (lastEvent = "no_event"; state = "PAUSE")
         ]
        )
       ]; "", TrackedSymbols → {tick}]],

     EventHandler[
      Dynamic[If[showPhase,
        Refresh[Grid[{{Framed[Show[phasePortraitPlot,
              Graphics[{Blue, PointSize[0.02], Point[{θ1[currentTime], θ1Speed[currentTime]}]}]],
             FrameStyle -> Directive[Thickness[.005], Gray]]}, {Framed[Graphics[
              {
               {RGBColor[{188, 143, 143} / 255], Circle[{0, 0}, L1]},
               {Circle[{0, 0}, 0.05]},
               getCoordinates[nBobs, bob1, bob2, bob3, m1, m2, m3]
              },
              graphicsOptions2D[6.2, {370, 312}]
```

```
        ], FrameStyle -> Directive[Thickness[.005], Gray]
       ]
      }}, Alignment → Center], TrackedSymbols → {tick, bob1, bob2, bob3}]
    ,
    Refresh[Framed[Graphics[
       {
        {RGBColor[{188, 143, 143} / 255], Circle[{0, 0}, L1]},
        {Circle[{0, 0}, 0.05]},

        getCoordinates[nBobs, bob1, bob2, bob3, m1, m2, m3]
       },
       graphicsOptions2D[6.2, {370, 497}]
      ], FrameStyle -> Directive[Thickness[.005], Gray]], TrackedSymbols → {tick, bob1, bob2, bob3}]
    ]
   ]
   ,
   {
    "MouseDown" :→
     (lastEvent = "mouseDown";
      {θ1InitMouse, θ2InitMouse, θ3InitMouse} = {θ1Init, θ2Init, θ3Init}
     ),

    "MouseDragged" :→
     (
      p1 = MousePosition["Graphics"];

      {bob1, bob2, bob3, θ1InitMouse, θ2InitMouse, θ3InitMouse} = obtainScreenPositions[
        p1, bob1, bob2, bob3, θ1InitMouse, θ2InitMouse, θ3InitMouse, L1, L2, L3, nBobs];
     ),
    "MouseUp" :→
     (
      currentTime = 0;
      lastEvent = "mouseup";
      p1 = MousePosition["Graphics"];
      {θ1Init, θ2Init, θ3Init} = {θ1InitMouse, θ2InitMouse, θ3InitMouse};
      tick += DEL
     )
   }
  ]
 }],
(*----------------- controls -------------------------------*)
Item[
 Grid[{

   {Grid[{(* BLOCK 1 *)
      {
       Button[Text[Style["play", 14]], (lastEvent = "run_button"; tick += DEL), ImageSize -> {74, 30}],
       Button[Text[Style["pause", 14]], (lastEvent = "pause_button"; tick += DEL), ImageSize -> {74, 30}]
      },
      {
       Button[Text[Style["step", 14]], (lastEvent = "step_button"; tick += DEL), ImageSize -> {74, 30}],
       Button[Text[Style["reset", 14]], (lastEvent = "reset_button"; tick += DEL), ImageSize -> {74, 30}]
      }
     }, Spacings -> {.8, .3}, Alignment -> Center
    ]
   },


   {Grid[{
      {Text@Style["duration", 12],

       Manipulator[Dynamic[maxRunTime, (maxRunTime = #; {lastEvent = "duration_changed", tick += DEL}; #) &],
```

```
      {1, 100, 1}, ImageSize -> Tiny, ContinuousAction -> False],
     Dynamic@Text@Style[padIt2[maxRunTime, {5, 0}], 10]},

    {Style[Row[{"Δ", Style["t", Italic]}], "TR", 12],
     Manipulator[Dynamic[dt, (dt = #; {lastEvent = "delt_changed", tick += DEL}; #) &], {0.01, 0.1, 0.01},
      ImageSize -> Tiny, ContinuousAction -> False], Dynamic@Text@Style[padIt2[dt, {3, 3}], 10]
    }
   }]
  },

  {Grid[{
     {Text@Style["number of bobs", 12],
      RadioButtonBar[Dynamic[nBobs, (nBobs = #; {lastEvent = "initial_conditions_changed", tick += DEL};
           #) &], {1 → Text@Style["1", 10], 2 -> Text@Style["2", 10], 3 -> Text@Style["3", 10]}]
     }}
    , Frame → None, Spacings → {0.2, 0.4}, FrameStyle -> Directive[Thickness[.005], Gray]
   ]
  },

  {Grid[{(*adjust mass*)

     {Button[ Text@Style["min", 10],
        (m1 = 1; lastEvent = "initial_conditions_changed"; tick += DEL), ImageSize → Small,
       Alignment → Bottom], Spacer[2], Text@Style[Subscript[Style["m", Italic], "1"], "TR", 10],
      Manipulator[Dynamic[m1, (m1 = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
       {1, 20, 0.01}, ImageSize -> Tiny, ContinuousAction -> False],
      Spacer[2], Dynamic@Text@Style[padIt2[m1, {5, 3}], 11]},

     {Button[ Text@Style["min", 10],
        (m2 = 1; lastEvent = "initial_conditions_changed"; tick += DEL), ImageSize → Small,
       Alignment → Bottom], Spacer[2], Text@Style[Subscript[Style["m", Italic], "2"], "TR", 10],
      Manipulator[Dynamic[m2, (m2 = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
       {1, 20, 0.01}, ImageSize -> Tiny, ContinuousAction -> False],
      Spacer[2], Dynamic@Text@Style[padIt2[m2, {5, 3}], 11]},

     {Button[ Text@Style["min", 10],
        (m3 = 1; lastEvent = "initial_conditions_changed"; tick += DEL), ImageSize → Small,
       Alignment → Bottom], Spacer[2], Text@Style[Subscript[Style["m", Italic], "3"], "TR", 10],
      Manipulator[Dynamic[m3, (m3 = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
       {1, 20, 0.01}, ImageSize -> Tiny, ContinuousAction -> False],
      Spacer[2], Dynamic@Text@Style[padIt2[m3, {5, 3}], 11]},

     {Button[ Text@Style["min", 10],
        (L1 = 1; lastEvent = "initial_conditions_changed"; tick += DEL), ImageSize → Small,
       Alignment → Bottom], Spacer[2], Text@Style[Subscript[Style["L", Italic], "1"], "TR", 10],
      Manipulator[Dynamic[L1, (L1 = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
       {1, 2, 0.1}, ImageSize -> Tiny, ContinuousAction -> False],
      Spacer[2], Dynamic@Text@Style[padIt2[L1, {3, 2}], 11]},

     {Button[ Text@Style["min", 10],
        (L2 = 1; lastEvent = "initial_conditions_changed"; tick += DEL), ImageSize → Small,
       Alignment → Bottom], Spacer[2], Text@Style[Subscript[Style["L", Italic], "2"], "TR", 10],
      Manipulator[Dynamic[L2, (L2 = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
       {1, 2, 0.1}, ImageSize -> Tiny, ContinuousAction -> False],
      Spacer[2], Dynamic@Text@Style[padIt2[L2, {3, 2}], 11]},

     {Button[ Text@Style["min", 10],
        (L3 = 1; lastEvent = "initial_conditions_changed"; tick += DEL), ImageSize → Small,
```

```
      Alignment → Bottom], Spacer[2], Text@Style[Subscript[Style["L", Italic], "3"], "TR", 10],
     Manipulator[Dynamic[L3, (L3 = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
      {1, 2, 0.1}, ImageSize -> Tiny, ContinuousAction -> False],
     Spacer[2], Dynamic@Text@Style[padIt2[L3, {3, 2}], 11]}

   }, Frame → None, Spacings → {0.2, 0}, Spacings → {0.2, 0}, Alignment → Left]
 },

 {Grid[{(*initial conditions*)

    {Button[ Text@Style["zero", 10], (θ1Init = 0; lastEvent = "initial_conditions_changed"; tick += DEL),
      ImageSize → Small, Alignment → Bottom], Spacer[2],
     Text@Style[Subscript["θ", "1"], "TR", 10], Manipulator[Dynamic[θ1Init,
       (θ1Init = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
      {-N@Pi, N@Pi, Pi/100.}, ImageSize -> Tiny, ContinuousAction -> False],
     Spacer[2], Dynamic@Text@Style[padIt1[θ1Init, {4, 1}], 10]},

    {Button[ Text@Style["zero", 10], (θ2Init = 0; lastEvent = "initial_conditions_changed"; tick += DEL),
      ImageSize → Small, Alignment → Bottom], Spacer[2],
     Text@Style[Subscript["θ", "2"], "TR", 10], Manipulator[Dynamic[θ2Init,
       (θ2Init = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
      {-N@Pi, N@Pi, Pi/100.}, ImageSize -> Tiny, ContinuousAction -> False],
     Spacer[2], Dynamic@Text@Style[padIt1[θ2Init, {4, 1}], 10]},

    {Button[ Text@Style["zero", 10], (θ3Init = 0; lastEvent = "initial_conditions_changed"; tick += DEL),
      ImageSize → Small, Alignment → Bottom], Spacer[2],
     Text@Style[Subscript["θ", "3"], "TR", 10], Manipulator[Dynamic[θ3Init,
       (θ3Init = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
      {-N@Pi, N@Pi, Pi/100.}, ImageSize -> Tiny, ContinuousAction -> False],
     Spacer[2], Dynamic@Text@Style[padIt1[θ3Init, {4, 1}], 10]},

    {Button[ Text@Style["zero", 10], (θ1SpeedInit = 0; lastEvent = "initial_conditions_changed";
        tick += DEL), ImageSize → Small, Alignment → Bottom], Spacer[2],
     Text@Style[Overscript[Subscript["θ", "1"], "•"], "TR", 10], Manipulator[
      Dynamic[θ1SpeedInit, (θ1SpeedInit = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
      {-3, 3, .1}, ImageSize -> Tiny, ContinuousAction -> False],
     Spacer[2], Dynamic@Text@Style[padIt1[θ1SpeedInit, {3, 2}], 10]},

    {Button[ Text@Style["zero", 10], (θ2SpeedInit = 0; lastEvent = "initial_conditions_changed";
        tick += DEL), ImageSize → Small, Alignment → Bottom], Spacer[2],
     Text@Style[Overscript[Subscript["θ", "2"], "•"], "TR", 10], Manipulator[
      Dynamic[θ2SpeedInit, (θ2SpeedInit = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
      {-3, 3, .1}, ImageSize -> Tiny, ContinuousAction -> False],
     Spacer[2], Dynamic@Text@Style[padIt1[θ2SpeedInit, {3, 2}], 10]},

    {Button[ Text@Style["zero", 10], (θ3SpeedInit = 0; lastEvent = "initial_conditions_changed";
        tick += DEL), ImageSize → Small, Alignment → Bottom], Spacer[2],
     Text@Style[Overscript[Subscript["θ", "3"], "•"], "TR", 10], Manipulator[
      Dynamic[θ3SpeedInit, (θ3SpeedInit = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
      {-3, 3, .1}, ImageSize -> Tiny, ContinuousAction -> False],
     Spacer[2], Dynamic@Text@Style[padIt1[θ3SpeedInit, {3, 2}], 10]}

   }, Frame → None, Spacings → {0.2, 0.0}, Alignment → Left]},

  {Dynamic[makePeKeChart[currentPE, currentKE]]}
 }, Frame → All, FrameStyle -> Directive[Thickness[.005], Gray], Spacings → {0.3, 0.5}, Alignment → Center]
, ControlPlacement → Left],
```

```
(*RIGHT PANEL*)
Item[
 Grid[{

   {Grid[{
      {Text@Style["gravity", 10]},

      {PopupMenu[Dynamic[g, (g = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
        {1.63 → Text@Style["moon", 12], 9.81 → Text@Style["earth", 11],
         274.68 → Text@Style["sun", 11]}, ImageSize → Tiny]
      }
     }, Frame → None, Alignment → Center, Spacings → {0, .6}]},

   {Grid[{
      {Text@Style["damping", 10]},

      {VerticalSlider[Dynamic[c, (c = #; {lastEvent = "initial_conditions_changed", tick += DEL}; #) &],
        {0, 3, .1}, ImageSize -> Tiny, ContinuousAction -> False]},

      {Dynamic@Text@Style[padIt2[c, {3, 1}], 10]}
     }, Frame → None, Alignment → Center, Spacings → {0, .6}]},

   {Grid[{
      {Text@Style["show phase", 10]},
      {Checkbox[Dynamic[showPhase, (showPhase = #; {lastEvent = "show_phase", tick += DEL}; #) &]]}
     }, Frame → None, Alignment → Center, Spacings → {0, .6}]},

   {Grid[{
      {Text@Style["current time", 10], Spacer[5]},
      {Dynamic@Text[Style[padIt2[currentTime, {5, 2}], 10]]}
     }, Frame → None, Alignment → Center, Spacings → {0, .6}]}

  }, Frame → All, FrameStyle -> Directive[Thickness[.005], Gray], Spacings → {0.3, 0.5}, Alignment → Center
 ], ControlPlacement → Right
],

{{currentPE, 1}, ControlType → None},
{{currentKE, 0}, ControlType → None},
{{showPhase, False}, None},
{{phasePortraitPlot, {}}, None},
{{g, 9.81}, None},
{{c, 0}, None},
{{maxRunTime, 100}, None},
{{dt, 0.05}, None},
{{bob1, {0, 0}}, ControlType → None},
{{bob2, {0, 0}}, ControlType → None},
{{bob3, {0, 0}}, ControlType → None},
{{θ1, 0}, None},
{{θ1Speed, 0}, None},
{{nBobs, 3}, None},
{{m1, 20}, None},
{{m2, 15}, None},
{{m3, 20}, None},
{{L1, 2}, None},
{{L2, 2}, None},
{{L3, 2}, None},
{{θ1InitMouse, 0}, None},
{{θ2InitMouse, 0}, None},
{{θ3InitMouse, 0}, None},
{{state, "RESET"}, None}, (*adjust this to RESET if do not want it to start in run mode*)
{{θ1Init, 0.6}, None},
{{θ2Init, 0.1}, None},
```

```
{{θ3Init, 0.1}, None},
{{θ1SpeedInit, 2.3}, None},
{{θ2SpeedInit, 0}, None},
{{θ3SpeedInit, 0}, None},
{{p1, 0}, None},
{{lastEvent, "no_event"}, None},
{{currentTime, 0}, None},
{{tick, 0}, None},
{{DEL, $MachineEpsilon}, None},

TrackedSymbols → {None},
ContinuousAction → False,
SynchronousUpdating → False,
SynchronousInitialization → False,
ControlPlacement → Left,
Alignment → Center,
ImageMargins → 0,
FrameMargins → 0,

Initialization :→
 (

 Off[InterpolatingFunction::dmval];

 (*formating functions*)
 (*---------------------------------------------------------*)
 padIt1[v_? (NumberQ[#] &), f_List] :=
  AccountingForm[Chop[N@v] , f, NumberSigns → {"-", "+"}, NumberPadding → {"0", "0"}, SignPadding → True
  ];

 (*---------------------------------------------------------*)
 padIt2[v_? (NumberQ[#] &), f_List] :=
  AccountingForm[Chop[N@v] , f, NumberSigns → {"", ""}, NumberPadding → {"0", "0"}, SignPadding → True
  ];

 (*---------------------------------------------------------*)
 graphicsOptions2D[maxExt_, imageSize_] := Module[{frameThickness = 0.001},
   {
    ImageSize → imageSize, ImagePadding → 2, ImageMargins → 0,
    Axes → False,
    PlotRange → {{-maxExt, maxExt }, {-maxExt , maxExt }},
    AspectRatio → 1,
    Background → White,
    TicksStyle → Small,
    PlotRangePadding → None,
    AxesStyle → Directive[{Blue, Thickness[frameThickness]}],
    AspectRatio → 1
   }
  ];

 (*---------------------------------------------------------*)
 getCoordinates[nBobs_, bob1_, bob2_, bob3_, m1_, m2_, m3_] :=
  Module[{slope = 0.03 / 19, c = 0.01 - (0.03 / 19)},

   Which[nBobs == 1,
    {
     {Red, Style[Line[{{0, 0}, bob1}], Antialiasing → True]},
     {Blue, PointSize[slope * m1 + c], Style[Point[bob1], Antialiasing → True]}
    },
    nBobs == 2,
    {
     {Red, Style[Line[{{0, 0}, bob1, bob2}], Antialiasing → True]},
     {Blue, PointSize[slope * m1 + c], Style[Point[bob1], Antialiasing → True]},
```

```
    {Blue, PointSize[slope*m2 + c], Style[Point[bob2], Antialiasing → True]}
   },
   True,
   {
    {Red, Style[Line[{{0, 0}, bob1, bob2, bob3}], Antialiasing → True]},
    {Blue, PointSize[slope*m1 + c], Style[Point[bob1], Antialiasing → True]},
    {Blue, PointSize[slope*m2 + c], Style[Point[bob2], Antialiasing → True]},
    {Blue, PointSize[slope*m3 + c], Style[Point[bob3], Antialiasing → True]}
   }
  ]
 ];

(*-------------------------------------------------------*)
obtainScreenPositions[p1_, bbob1_, bbob2_, bbob3_, θθ1Init_, θθ2Init_, θθ3Init_, L1_, L2_, L3_, nBobs_] :=
 Module[{bob1 = bbob1, bob2 = bbob2, bob3 = bbob3, θ1InitMouse = θθ1Init, θ2InitMouse = θθ2Init,
   θ3InitMouse = θθ3Init, delx, dely, oldbob2, newbob1, newbob2, newbob3},

  Which[nBobs == 1,
   (
    bob1 = positionOfNewBob[p1, L1];
    θ1InitMouse = normalizedAngleFromMouseInput[ArcTan[bob1[[1]], bob1[[2]]]]
   ),
   nBobs == 2,
   (
    If[EuclideanDistance[p1, bob1] < EuclideanDistance[p1, bob2],
     (
      newbob1 = positionOfNewBob[p1, L1];
      delx = newbob1[[1]] - bob1[[1]];
      dely = newbob1[[2]] - bob1[[2]];
      bob1 = newbob1;
      bob2 = {bob2[[1]] + delx, bob2[[2]] + dely};

      θ1InitMouse = normalizedAngleFromMouseInput[ArcTan[bob1[[1]], bob1[[2]]]];
      θ2InitMouse = normalizedAngleFromMouseInput[ArcTan[bob2[[1]] - bob1[[1]], bob2[[2]] - bob1[[2]]]]
     )
     ,
     (
      bob2 = positionOfNewBob[p1, bob1, L2];
      θ2InitMouse = normalizedAngleFromMouseInput[ArcTan[bob2[[1]] - bob1[[1]], bob2[[2]] - bob1[[2]]]]
     )
    ]
   ),
   True,
   (
    If[EuclideanDistance[p1, bob1] < EuclideanDistance[p1, bob2],
     (
      If[EuclideanDistance[p1, bob1] < EuclideanDistance[p1, bob3],
       ( (*bob1*)

        newbob1 = positionOfNewBob[p1, L1];
        delx = newbob1[[1]] - bob1[[1]];
        dely = newbob1[[2]] - bob1[[2]];
        bob1 = newbob1;
        oldbob2 = bob2;
        bob2 = {bob2[[1]] + delx, bob2[[2]] + dely};
        θ1InitMouse = normalizedAngleFromMouseInput[ArcTan[bob1[[1]], bob1[[2]]]];

        θ2InitMouse =
         normalizedAngleFromMouseInput[ArcTan[bob2[[1]] - bob1[[1]], bob2[[2]] - bob1[[2]]]];

        delx = bob2[[1]] - oldbob2[[1]];
        dely = bob2[[2]] - oldbob2[[2]];
        bob3 = {bob3[[1]] + delx, bob3[[2]] + dely};
        θ3InitMouse =
```

```
          normalizedAngleFromMouseInput[ArcTan[bob3[[1]] - bob2[[1]], bob3[[2]] - bob2[[2]]]]
        ),
        ((*bob3*)
         newbob3 = positionOfNewBob[p1, bob2, L3];
         delx = newbob3[[1]] - bob3[[1]];
         dely = newbob3[[2]] - bob3[[2]];
         bob3 = {bob3[[1]] + delx, bob3[[2]] + dely};
         θ3InitMouse =
          normalizedAngleFromMouseInput[ArcTan[bob3[[1]] - bob2[[1]], bob3[[2]] - bob2[[2]]]]
        )
       ]
      )
      ,
      (
       If[EuclideanDistance[p1, bob2] < EuclideanDistance[p1, bob3],
        ( (*bob2*)

         newbob2 = positionOfNewBob[p1, bob1, L2];
         delx = newbob2[[1]] - bob2[[1]];
         dely = newbob2[[2]] - bob2[[2]];
         bob2 = {bob2[[1]] + delx, bob2[[2]] + dely};
         θ2InitMouse =
          normalizedAngleFromMouseInput[ArcTan[bob2[[1]] - bob1[[1]], bob2[[2]] - bob1[[2]]]];

         bob3 = {bob3[[1]] + delx, bob3[[2]] + dely};
         θ3InitMouse =
          normalizedAngleFromMouseInput[ArcTan[bob3[[1]] - bob2[[1]], bob3[[2]] - bob2[[2]]]]
        ),
        ((*bob3*)
         newbob3 = positionOfNewBob[p1, bob2, L3];
         delx = newbob3[[1]] - bob3[[1]];
         dely = newbob3[[2]] - bob3[[2]];
         bob3 = {bob3[[1]] + delx, bob3[[2]] + dely};
         θ3InitMouse =
          normalizedAngleFromMouseInput[ArcTan[bob3[[1]] - bob2[[1]], bob3[[2]] - bob2[[2]]]]
        )
       ]
      )]
   )
  ];

  {bob1, bob2, bob3, θ1InitMouse, θ2InitMouse, θ3InitMouse}
 ];

(* keep angle in range 0..Pi, -Pi...0 *)
(*---------------------------------------------------------*)
normalizedAngleFromMouseInput[θ_] := Module[{},

  Which[θ ≥ 0 && θ ≤ Pi / 2, θ + Pi / 2,
   θ > Pi / 2 && θ ≤ Pi, -(Pi / 2 + (Pi - θ)),
   True, Pi / 2 + θ
  ]
 ];

(*Helper function to determine which bob to move on the screen*)
(*when using eventHandler*)
(*---------------------------------------------------------*)
positionOfNewBob[p1_, L1_] := Module[{newBob, y, x, eq, pt},

  eq = y - p1[[2]] == (-p1[[2]] / -p1[[1]]) (x - p1[[1]]);
  pt = NSolve[{x^2 + y^2 == L1^2, eq}, {x, y}];
```



```
  If[EuclideanDistance[p1, {x /. pt[[1]], y /. pt[[1]]}] <
    EuclideanDistance[p1, {x /. pt[[2]], y /. pt[[2]]}],
   (
    newBob = {x /. pt[[1]], y /. pt[[1]]}
   ),
   (
    newBob = {x /. pt[[2]], y /. pt[[2]]}
   )
  ];

  newBob
 ];

(*---------------------------------------------------------*)
positionOfNewBob[p1_, bob_, len_] := Module[{newBob, y, x, eq, pt},

  eq = y - p1[[2]] == (bob[[2]] - p1[[2]]) / (bob[[1]] - p1[[1]]) (x - p1[[1]]);

  pt = NSolve[{(x - bob[[1]])^2 + (y - bob[[2]])^2 == len^2, eq}, {x, y}];

  If[EuclideanDistance[p1, {x /. pt[[1]], y /. pt[[1]]}] <
    EuclideanDistance[p1, {x /. pt[[2]], y /. pt[[2]]}],
   (
    newBob = {x /. pt[[1]], y /. pt[[1]]}
   ),
   (
    newBob = {x /. pt[[2]], y /. pt[[2]]}
   )
  ];

  newBob
 ];

(*---------------------------------------------------------*)
solve[nBobs_, θ1Init_, θ2Init_, θ3Init_, θ1SpeedInit_, θ2SpeedInit_,
  θ3SpeedInit_, m1_, m2_, m3_, L1_, L2_, L3_, g_, from_, to_, c_, accuracyGoal_] :=
 Module[{numericalSolution, ic, t, ndsolveOptions, x1, x2, x3, x1der, x2der, x3der},

  ndsolveOptions = {MaxSteps -> Infinity, Method ->
     {"StiffnessSwitching", Method -> {"ExplicitRungeKutta", Automatic}}, AccuracyGoal → accuracyGoal};

  Which[
   nBobs == 1,
   (
    ic = {x1[0] == θ1Init , x1'[0] == θ1SpeedInit};
    numericalSolution =
     First@NDSolve[
       Flatten[{eqOneBob[L1, m1, g, t, x1, c] == 0, ic}],
       {x1, x1'}, {t, from, to}, Sequence@ndsolveOptions];

    x1 = x1 /. numericalSolution;
    x1der = x1' /. numericalSolution
   ),
   nBobs == 2,
   (
    ic = {x1[0] == θ1Init , x1'[0] == θ1SpeedInit, x2[0] == θ2Init , x2'[0] == θ2SpeedInit};
    numericalSolution =
     First@NDSolve[
       Flatten[{
         eqOne2Bob[L1, L2, m1, m2, g, t, x1, x2, c] == 0,
         eqTwo2Bob[L1, L2, m2, g, t, x1, x2, c] == 0, ic}],
```

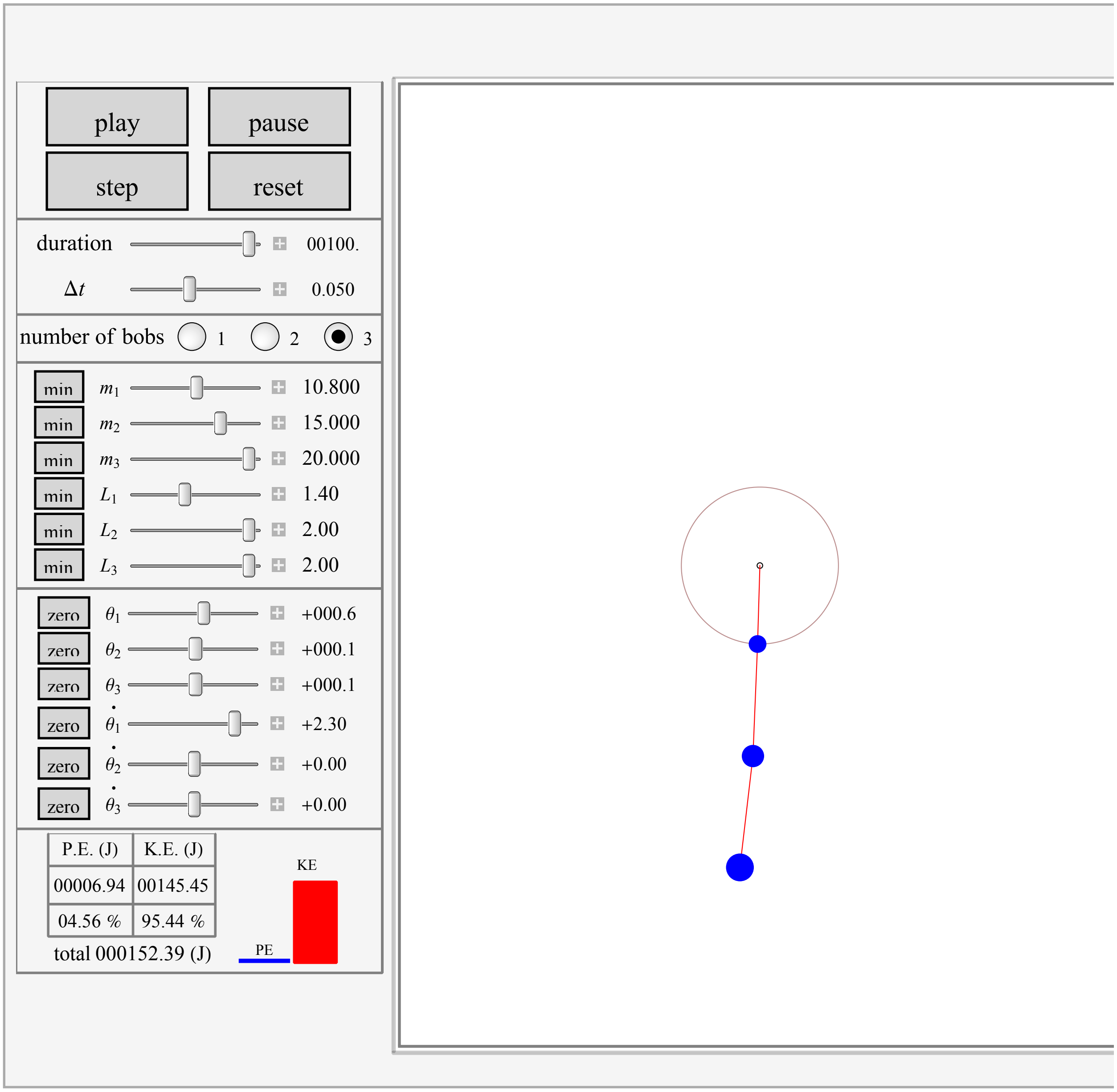
```
        {x1, x1', x2, x2'}, {t, from, to}, Sequence@ndsolveOptions];
     {x1, x2, x1der, x2der} = {x1 /. numericalSolution, x2 /. numericalSolution,
       x1' /. numericalSolution, x2' /. numericalSolution}
    ),
    True,
    (
     ic = {x1[0] == θ1Init , x1'[0] == θ1SpeedInit,
       x2[0] == θ2Init , x2'[0] == θ2SpeedInit, x3[0] == θ3Init , x3'[0] == θ3SpeedInit};

     numericalSolution = First@NDSolve[
        Flatten[{
          eqOne3Bob[L1, L2, L3, m1, m2, m3, g, t, x1, x2, x3, c] == 0,
          eqTwo3Bob[L1, L2, L3, m2, m3, g, t, x1, x2, x3, c] == 0,
          eqThree3Bob[L1, L2, L3, m3, g, t, x1, x2, x3, c] == 0,
          ic}],
        {x1, x1', x2, x2', x3, x3'},
        {t, from, to},
        Sequence@ndsolveOptions];

     {x1, x2, x3, x1der, x2der, x3der} = {x1 /. numericalSolution, x2 /. numericalSolution, x3 /.
        numericalSolution, x1' /. numericalSolution, x2' /. numericalSolution, x3' /. numericalSolution}
    )
   ];

   {x1, x2, x3, x1der, x2der, x3der}
  ];

(*--------------------------------------------------------*)
makePhasePortrait[nBobs_, θ1Init_, θ2Init_, θ3Init_, θ1SpeedInit_, θ2SpeedInit_, θ3SpeedInit_,
   m1_, m2_, m3_, L1_, L2_, L3_, g_, c_, maxRunTime_, dt_] := Module[{θ1, θ2, θ3, θ1Speed,
    θ2Speed, θ3Speed, data, imageSize = {365, 161}, t, icPoint, finalPoint, accuracyGoal = 4},


   {θ1, θ2, θ3, θ1Speed, θ2Speed, θ3Speed} = solve[nBobs, θ1Init, θ2Init, θ3Init, θ1SpeedInit,
     θ2SpeedInit, θ3SpeedInit, m1, m2, m3, L1, L2, L3, g, 0, maxRunTime, c, accuracyGoal];

   data = Table[{θ1[t], θ1Speed[t]}, {t, 0, maxRunTime, dt}];
   icPoint = {{Black, Text[Style["(i.c)", 12], {θ1Init, θ1SpeedInit}, {0, 1.5}]},
     {Red, PointSize[0.02], Point[{θ1Init, θ1SpeedInit}]}};
   finalPoint = {Black, PointSize[0.02], Point[{θ1[maxRunTime], θ1Speed[maxRunTime]}]};

   Show[ListPlot[data,
     Joined → True,
     PlotStyle → {RGBColor[{250, 128, 114} / 255], Directive[Thickness[0.001]]},
     ImageSize → imageSize,
     PlotRange → All,
     ImageMargins → 1,
     AspectRatio → .3,
     TicksStyle → Directive[7],
     AxesLabel → {
       Text@Grid[{{Style["θ", 10]}, {Style["(rad)", 10]}}, Alignment → Center, Spacings → {0, 0}],
       Text@Style[" θ̇ (rad/sec)", 10]
      }
    ], Graphics@icPoint, Graphics@finalPoint, PlotRange → All]

  ];

(*Below are the ode equestions. For different number of bobs*)
(*I derived these in a different notebook using basic Lagrangian derivation*)
(*--------------------------------------------------------*)
```

```
eqOneBob[L1_, m1_, g_, t_, x1_, c_] := Module[{},
  c x1′[t] + L1 m1 (g Sin[x1[t]] + L1 x1′′[t])
 ];

(*-----------------------------------------------------------*)
eqOne2Bob[L1_, L2_, m1_, m2_, g_, t_, x1_, x2_, c_] := Module[{},
  c x1 '[t] + L2 m2 Sin[x1[t] - x2[t]] x2′[t]^2 +
   (m1 + m2) (g Sin[x1[t]] + L1 x1′′[t]) + L2 m2 Cos[x1[t] - x2[t]] x2′′[t]
 ];

(*-----------------------------------------------------------*)
eqTwo2Bob[L1_, L2_, m2_, g_, t_, x1_, x2_, c_] := Module[{},
  c x2 '[t] + m2 (g Sin[x2[t]] - L1 Sin[x1[t] - x2[t]] x1′[t]^2 + L1 Cos[x1[t] - x2[t]] x1′′[t] + L2 x2′′[t])
 ];

(*-----------------------------------------------------------*)
eqOne3Bob[L1_, L2_, L3_, m1_, m2_, m3_, g_, t_, x1_, x2_, x3_, c_] := Module[{}, c x1 '[t] +
   g (m1 + m2 + m3) Sin[x1[t]] + L2 (m2 + m3) Sin[x1[t] - x2[t]] x2′[t]^2 + L3 m3 Sin[x1[t] - x3[t]] x3′[t]^2 +
   L1 m1 x1′′[t] + (m2 + m3) (L1 x1′′[t] + L2 Cos[x1[t] - x2[t]] x2′′[t]) + L3 m3 Cos[x1[t] - x3[t]] x3′′[t]
 ];

(*-----------------------------------------------------------*)
eqTwo3Bob[L1_, L2_, L3_, m2_, m3_, g_, t_, x1_, x2_, x3_, c_] := Module[{},
  c x2 '[t] - L1 (m2 + m3) Sin[x1[t] - x2[t]] x1′[t]^2 + L3 m3 Sin[x2[t] - x3[t]] x3′[t]^2 +
   (m2 + m3) (g Sin[x2[t]] + L1 Cos[x1[t] - x2[t]] x1′′[t] + L2 x2′′[t]) + L3 m3 Cos[x2[t] - x3[t]] x3′′[t]
 ];

(*-----------------------------------------------------------*)
eqThree3Bob[L1_, L2_, L3_, m3_, g_, t_, x1_, x2_, x3_, c_] := Module[{},
  c x3 '[t] + m3 (g Sin[x3[t]] - L1 Sin[x1[t] - x3[t]] x1′[t]^2 - L2 Sin[x2[t] - x3[t]] x2′[t]^2 +
      L1 Cos[x1[t] - x3[t]] x1′′[t] + L2 Cos[x2[t] - x3[t]] x2′′[t] + L3 x3′′[t])
 ];

(*-----------------------------------------------------------*)
updatePendulumPositions[nBobs_, θ1_, θ2_, θ3_, L1_, L2_, L3_] := Module[{bob1 = 0, bob2 = 0, bob3 = 0},
  (*check how many bobs we are using, and update the solution*)

  Which[nBobs == 1,
   (
    bob1 = {L1 Sin[θ1], -L1 Cos[θ1]}
   ),

   nBobs == 2,
   (
    bob1 = {L1 Sin[θ1], -L1 Cos[θ1]};
    bob2 = bob1 + {L2 Sin[θ2], -L2 Cos[θ2]}
   ),
   True,
   (bob1 = {L1 Sin[θ1], -L1 Cos[θ1]};
    bob2 = bob1 + {L2 Sin[θ2], -L2 Cos[θ2]};
    bob3 = bob2 + {L3 Sin[θ3], -L3 Cos[θ3]}
   )
  ];

  {bob1, bob2, bob3}
 ];

(*-----------------------------------------------------------*)
(*update bob positions when IC changes*)
resetUsingInitialConditions[nBobs_, θ1Init_, θ2Init_,
  θ3Init_, θ1SpeedInit_, θ2SpeedInit_, θ3SpeedInit_, bbob2_, bbob3_, L1_, L2_, L3_] :=
```

```
 Module[{θ1, θ2, θ3, θ1Speed, θ2Speed, θ3Speed, bob1, bob2 = bbob2, bob3 = bbob3},

  θ1 = θ1Init * Pi / 180.;
  θ2 = θ2Init * Pi / 180.;
  θ3 = θ3Init * Pi / 180.;
  θ1Speed = θ1SpeedInit;
  θ2Speed = θ2SpeedInit;
  θ3Speed = θ3SpeedInit;

  Which[nBobs == 1,
   (
    bob1 = {L1 Sin[θ1], -L1 Cos[θ1]}
   ),
   nBobs == 2,
   (
    bob1 = {L1 Sin[θ1], -L1 Cos[θ1]};
    bob2 = bob1 + {L2 Sin[θ2], -L2 Cos[θ2]}
   ),
   True,
   (
    bob1 = {L1 Sin[θ1], -L1 Cos[θ1]};
    bob2 = bob1 + {L2 Sin[θ2], -L2 Cos[θ2]};
    bob3 = bob2 + {L3 Sin[θ3], -L3 Cos[θ3]}
   )
  ];

  {θ1, θ2, θ3, θ1Speed, θ2Speed, θ3Speed, bob1, bob2, bob3}
 ];
(*--------------------------------------------------------*)
makePeKeChart[currentPE_, currentKE_] :=
 Module[{g1, peValueAsPercentage, keValueAsPercentage, totalE},

  totalE = currentPE + currentKE;

  If[totalE ≤ $MachineEpsilon,
   (
    peValueAsPercentage = 0;
    keValueAsPercentage = 0

   ),
   (
    peValueAsPercentage = Abs[currentPE] / totalE * 100;
    keValueAsPercentage = currentKE / totalE * 100
   )
  ];

  g1 = Grid[{
     {
      Text@Style["P.E. (J)", 10], Text@Style["K.E. (J)", 10]
     },
     {
      Text@Row[{Style[padIt2[currentPE, {7, 2}], 10]}],
      Text@Row[{Style[padIt2[currentKE, {7, 2}], 10]}]
     },
     {
      Text@Row[{Style[padIt2[peValueAsPercentage, {4, 2}], 10], Style[" %", 10]}],

      Text@Row[{Style[padIt2[keValueAsPercentage, {4, 2}], 10], Style[" %", 10]}]
     }


    }, Frame → All,
    FrameStyle -> Directive[Thickness[.005], Gray], Spacings → {0.4, 0.2}, Alignment → Center
   ];
```

```
  Grid[{
    {Grid[{
       {g1},
       {Text@Row[{Style["total ", 11], Style[padIt2[totalE, {8, 2}], 11], Style[" (J)", 11]}]}
      }, Spacings → {0, 0}, Alignment → Center], Graphics[
      {
       {Blue, Rectangle[{0, 0}, {60, peValueAsPercentage}]},
       {Red, Rectangle[{65, 0}, {115, keValueAsPercentage}]},
       Text[Style["PE", 8], {30, 1.1 peValueAsPercentage}, {0, -1}],
       Text[Style["KE", 8], {80, 1.1 * keValueAsPercentage}, {0, -1}]
      }, ImageSize → {70, 63}, ImageMargins → 0, ImagePadding → 0, PlotRange → {{-20, 125}, {-1, 140}}
     ]
    }
   }, Frame → None, Spacings → {0, 0}, Alignment → Center
  ]

 ];

(*Calulate PE*)
(*-------------------------------------------------------*)
pe[nBobs_, m1_, mm2_, mm3_, L1_, LL2_, LL3_, g_, θ1_, θθ2_, θθ3_] := Module[{m2, m3, L2, L3, θ2, θ3},

  Which[
   nBobs == 1, (m2 = 0; m3 = 0; L2 = 0; L3 = 0; θ2 = 0; θ3 = 0),
   nBobs == 2, (m2 = mm2; L2 = LL2; m3 = 0; L3 = 0; θ2 = θθ2; θ3 = 0),
   True, (m2 = mm2; L2 = LL2; θ2 = θθ2; m3 = mm3; L3 = LL3; θ3 = θθ3);
  ];

  g L1 m1 (1 - Cos[θ1]) + g m2 (L1 (1 - Cos[θ1]) + L2 (1 - Cos[θ2])) +
   g m3 (L1 (1 - Cos[θ1]) + L2 (1 - Cos[θ2]) + L3 (1 - Cos[θ3]))
 ];

(*calculate K.E.*)
(*-------------------------------------------------------*)
ke[nBobs_, θ1_, θθ2_, θθ3_, θ1Speed_, θθ2Speed_, θθ3Speed_, m1_, mm2_, mm3_, L1_, LL2_, LL3_] :=
 Module[{m2, m3, L2, L3, θ2, θ3, θ2Speed, θ3Speed},

  Which[

   nBobs == 1,
   (m2 = 0; m3 = 0; L2 = 0; L3 = 0; θ2Speed = 0; θ3Speed = 0; θ2 = 0; θ3 = 0),

   nBobs == 2,
   (m2 = mm2; L2 = LL2; θ2 = θθ2; θ2Speed = θθ2Speed; m3 = 0; L3 = 0; θ3 = 0; θ3Speed = 0),

   True,
   (m2 = mm2; L2 = LL2; θ2 = θθ2; θ2Speed = θθ2Speed; m3 = mm3; L3 = LL3; θ3Speed = θθ3Speed; θ3 = θθ3)
  ];

  1/2 m1 (L1 θ1Speed) ^ 2 +
   1/2 m2 ( (L1 θ1Speed Cos[θ1] + L2 θ2Speed Cos[θ2]) ^ 2 + (L1 θ1Speed Sin[θ1] + L2 θ2Speed Sin[θ2]) ^ 2) +
   1/2 m3 ( (L1 θ1Speed Cos[θ1] + L2 θ2Speed Cos[θ2] + L3 θ3Speed Cos[θ3]) ^ 2 +
      (L1 θ1Speed Sin[θ1] + L2 θ2Speed Sin[θ2] + L3 θ3Speed Sin[θ3]) ^ 2)
 ];

(*-------------------------------------------------------*)
update[nBobs_, θ1Init_, θ2Init_, θ3Init_, θ1SpeedInit_, θ2SpeedInit_, θ3SpeedInit_, m1_, m2_, m3_, L1_,
```

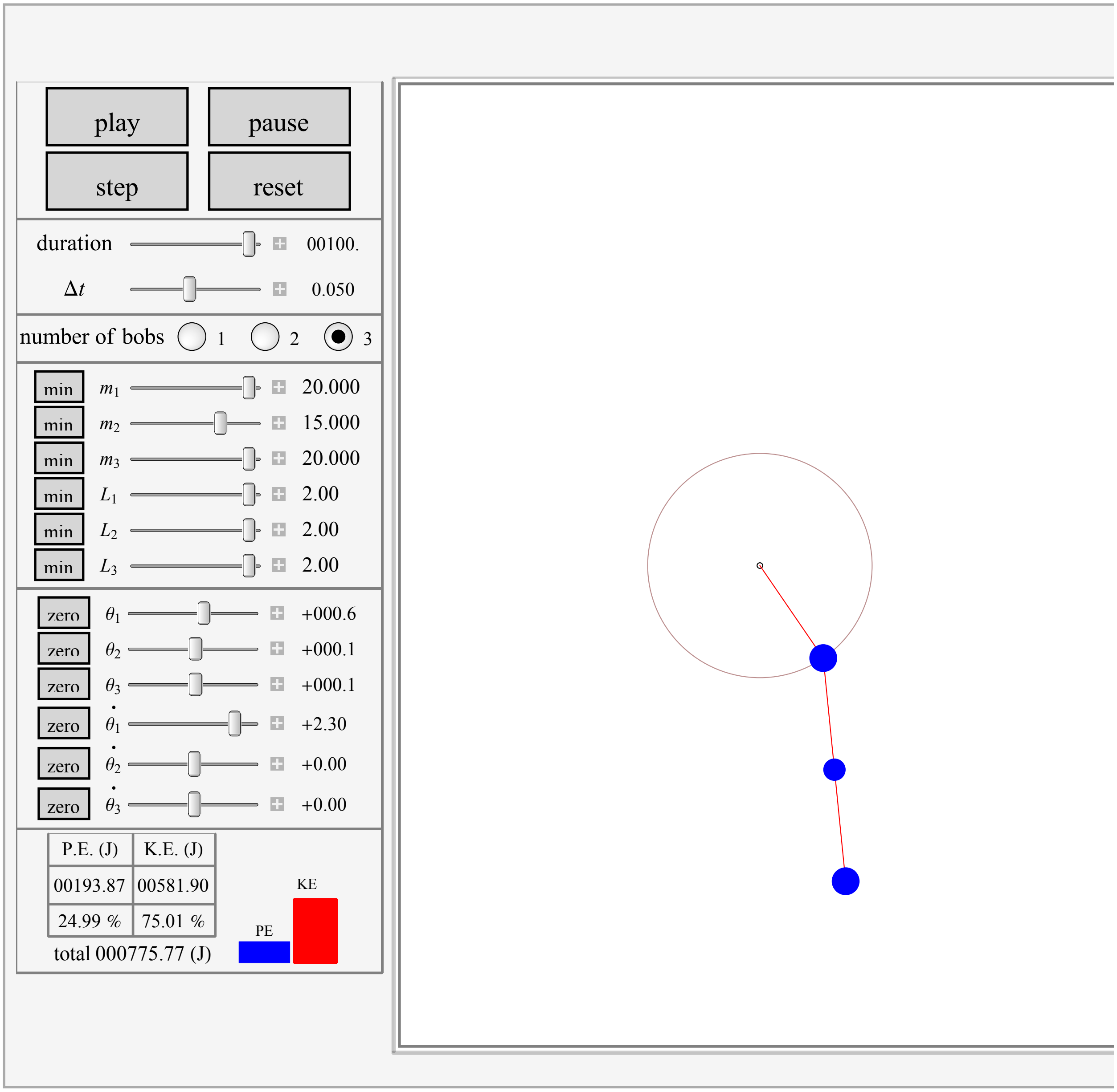
```
L2_, L3_, g_, currentTime_, c_, maxRunTime_, dt_, showPhase_] := Module[{currentPE, currentKE, bob1,
 bob2, bob3, θ1, θ2, θ3, θ1Speed, θ2Speed, θ3Speed, phasePortraitPlot = {}, accuracyGoal = 7},

{θ1, θ2, θ3, θ1Speed, θ2Speed, θ3Speed} = solve[nBobs, θ1Init, θ2Init, θ3Init, θ1SpeedInit, θ2SpeedInit,
  θ3SpeedInit, m1, m2, m3, L1, L2, L3, g, currentTime, currentTime + 1, c, accuracyGoal];

currentPE = pe[nBobs, m1, m2, m3, L1, L2, L3, g, θ1[currentTime], θ2[currentTime], θ3[currentTime]];
currentKE = ke[nBobs, θ1[currentTime], θ2[currentTime], θ3[currentTime],
  θ1Speed[currentTime], θ2Speed[currentTime], θ3Speed[currentTime], m1, m2, m3, L1, L2, L3];

If[showPhase,
 phasePortraitPlot = makePhasePortrait[nBobs, θ1Init, θ2Init, θ3Init,
   θ1SpeedInit, θ2SpeedInit, θ3SpeedInit, m1, m2, m3, L1, L2, L3, g, c, maxRunTime, dt]
];

{bob1, bob2, bob3} =
 updatePendulumPositions[nBobs, θ1[currentTime], θ2[currentTime], θ3[currentTime], L1, L2, L3];

{currentPE, currentKE, phasePortraitPlot, bob1, bob2, bob3, θ1, θ1Speed}

]
```



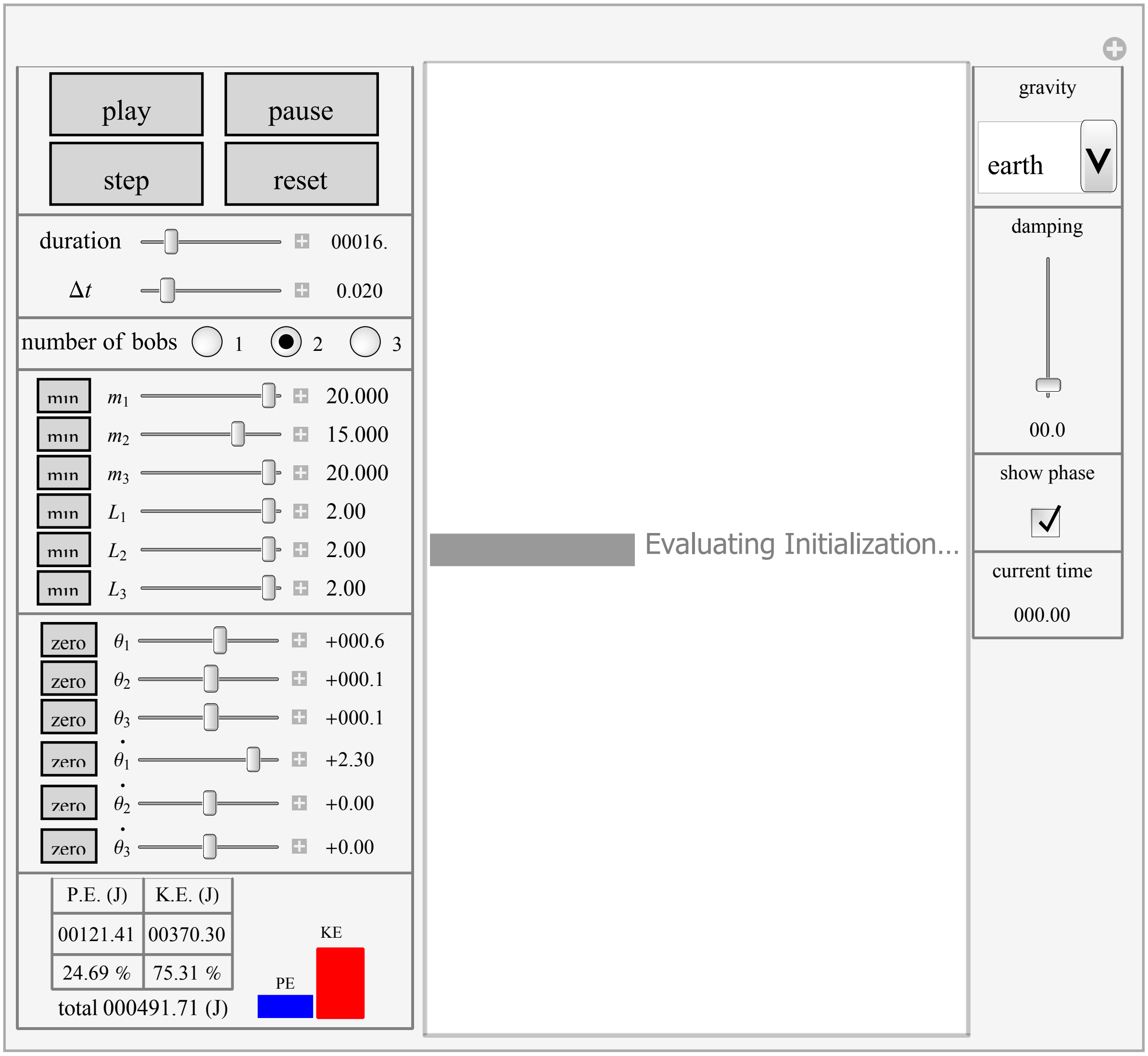
Caption

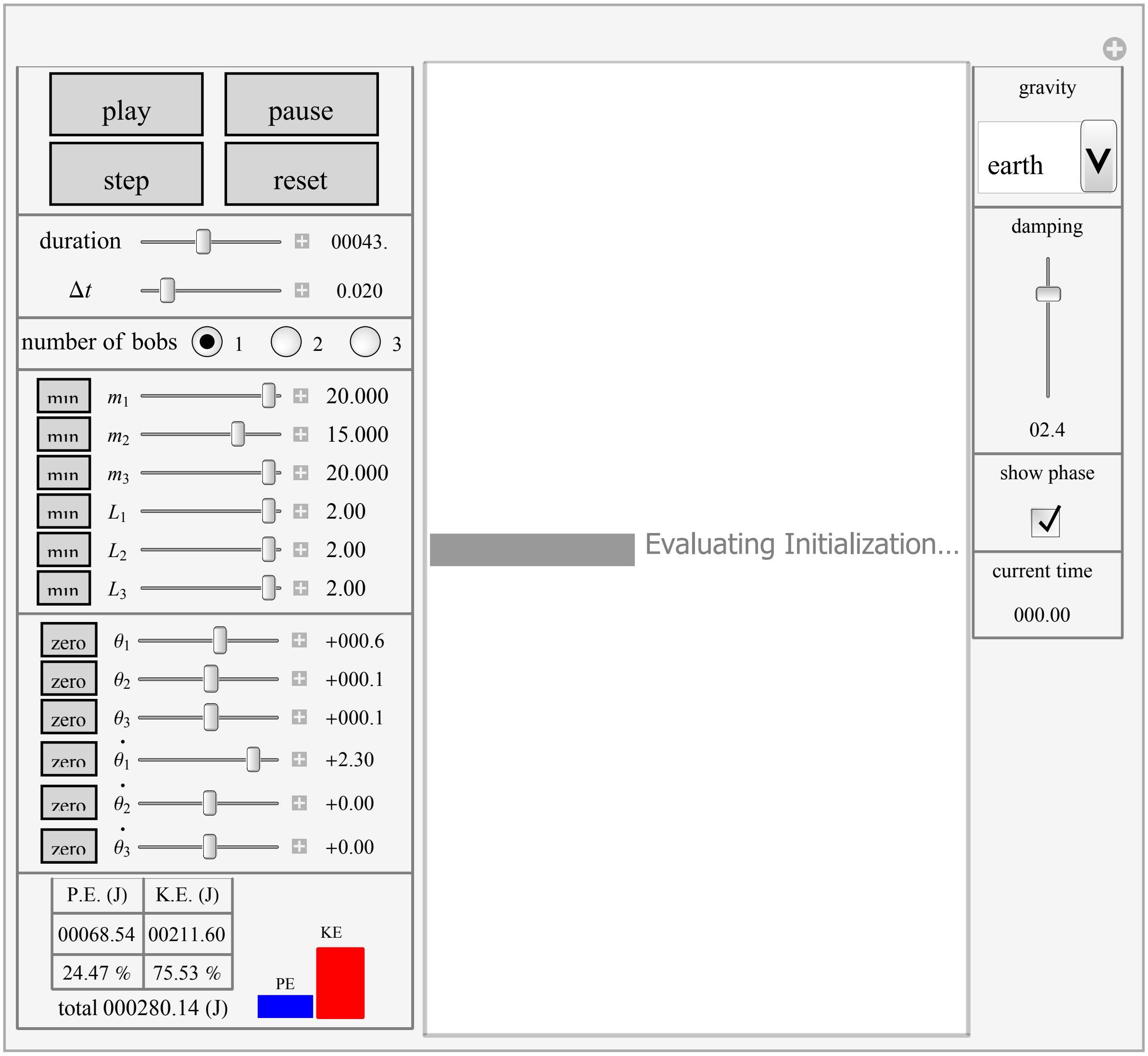
This Demonstration provides basic motion simulation of free moving damped triple pendulum. It can also be used as a double pendulum or a simple pendulum since the number of bobs is a configurable parameter. Viscous damping is due to the pendulums moving in fluid. Phase portrait diagram is updated during simulation. Energy of the system is displayed as the simulation is running and is seen to be constant as expected when damping is not present. Initial conditions for the positions of the pendulum bobs can be adjusted either using the mouse (by dragging a bob to a new location) or by using the control slides.
For double and triple pendulum, the phase portrait is applied to the inner most bob.

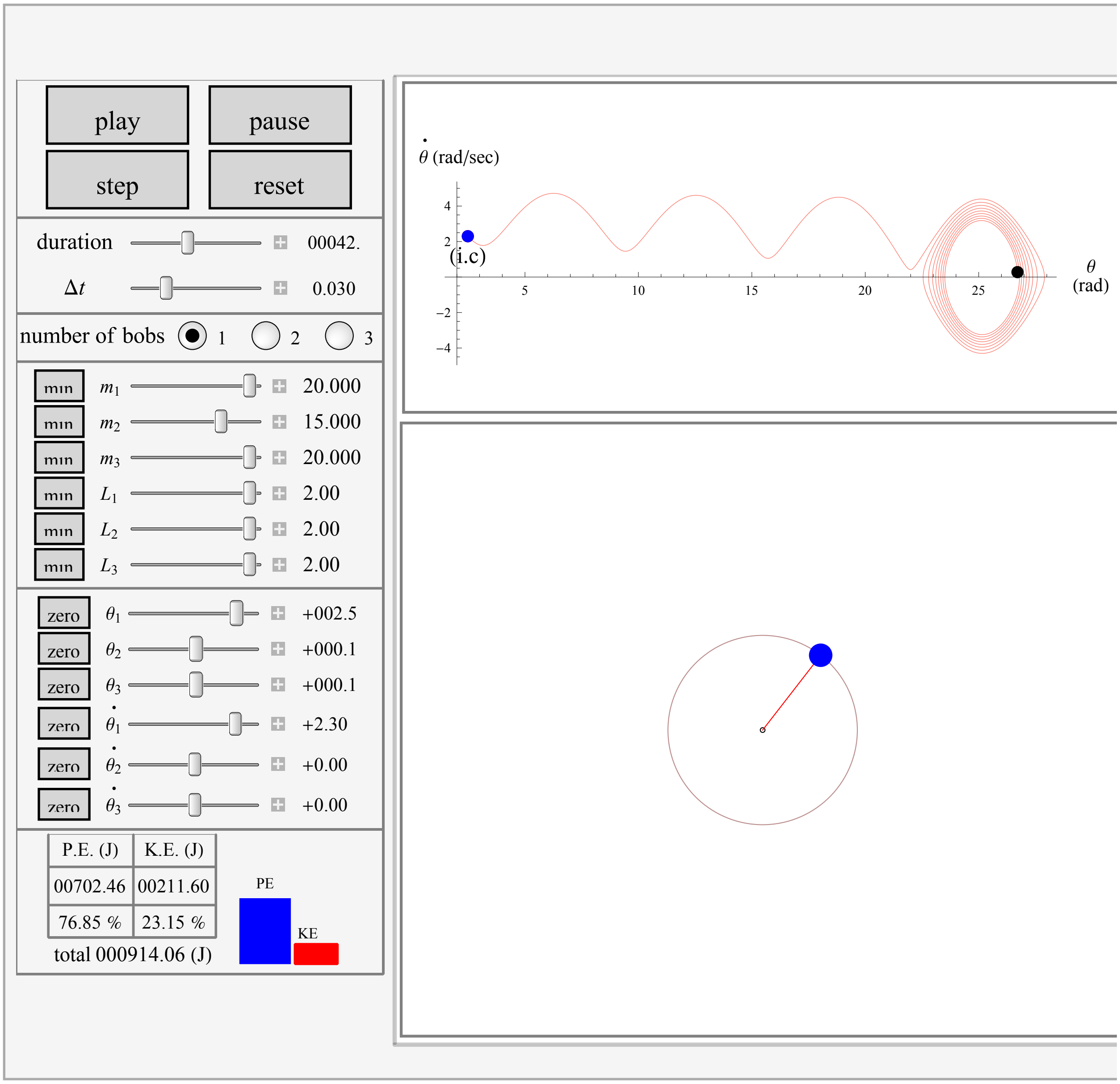
Thumbnail

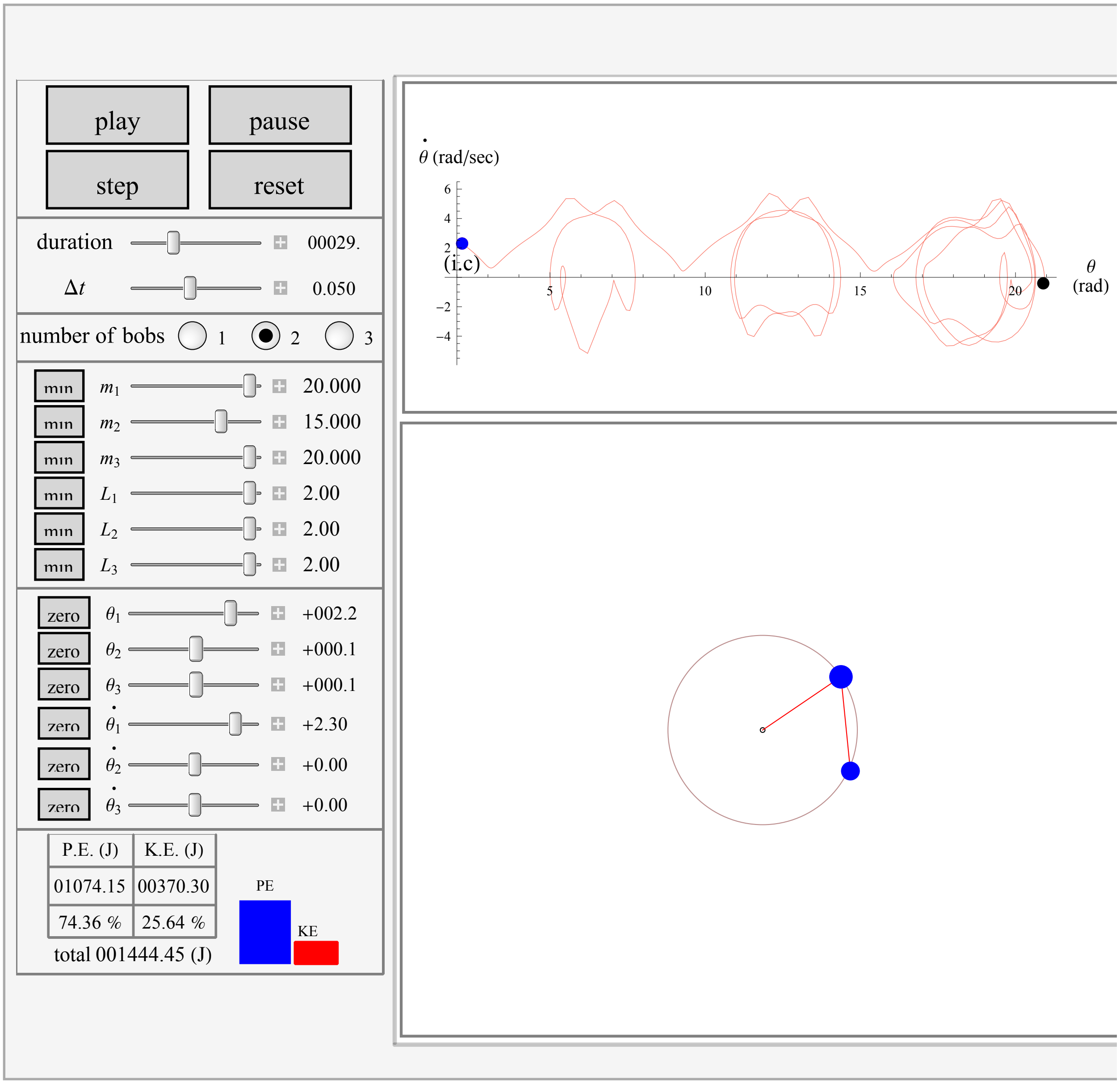

play
pause
step
reset
duration 00100.
Δt 0.050
number of bobs 1 2 3
min m_1 20.000
min m_2 15.000
min m_3 20.000
min L_1 2.00
min L_2 2.00
min L_3 2.00
zero θ_1 +000.6
zero θ_2 +000.1
zero θ_3 +000.1
zero $\dot{\theta}_1$ +2.30
zero $\dot{\theta}_2$ +0.00
zero $\dot{\theta}_3$ +0.00
P.E. (J) K.E. (J)
00193.87 00581.90
24.99 % 75.01 %
total 000775.77 (J)
KE
PE

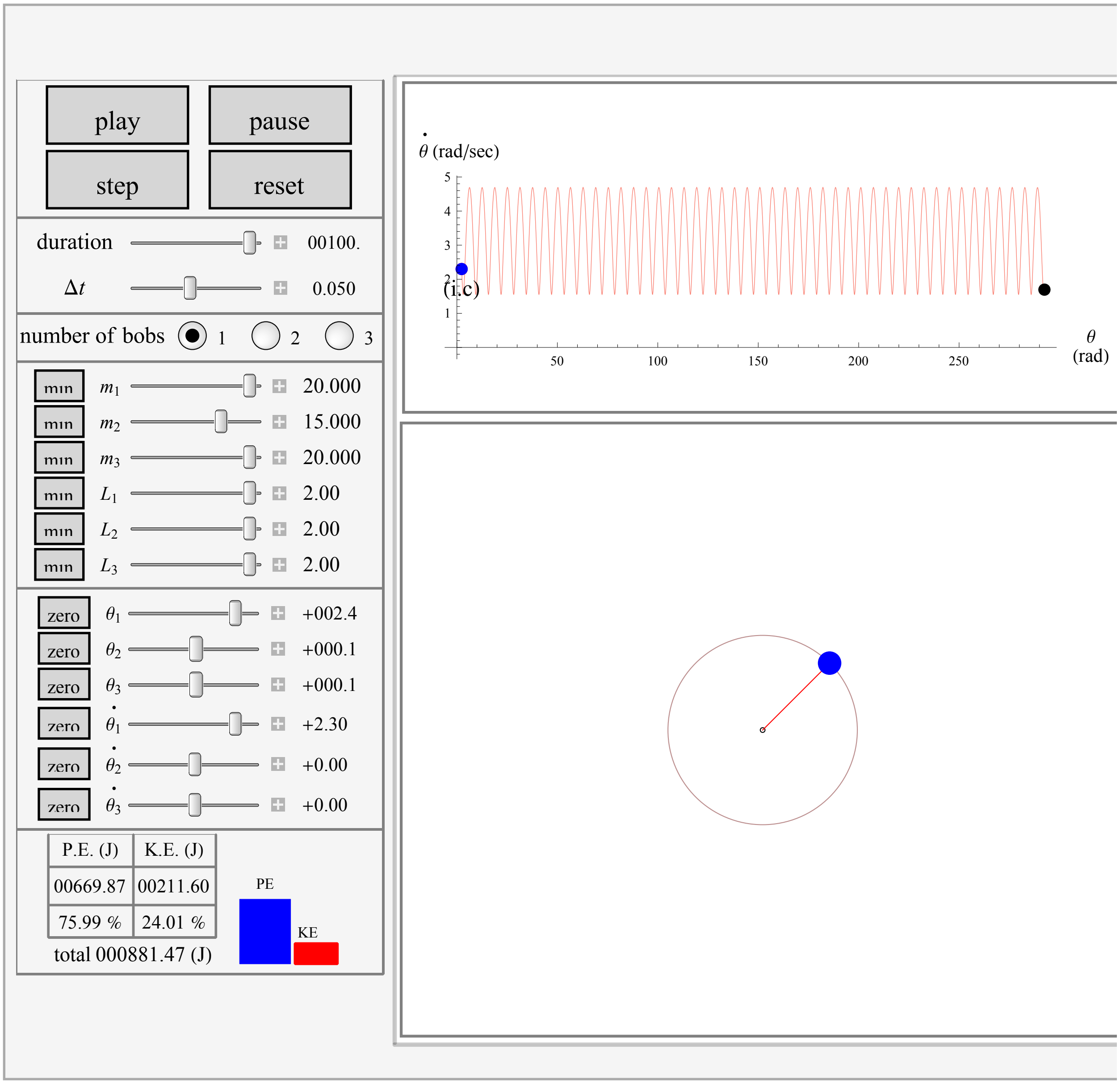
Snapshots


play
pause
step
reset
duration
00016.
Δt
0.020
number of bobs 1 2 3
min
m1
20.000
m2
15.000
m3
20.000
L1
2.00
L2
2.00
L3
2.00
zero
θ1
+000.6
θ2
+000.1
θ3
+000.1
+2.30
+0.00
+0.00
P.E. (J)
K.E. (J)
00121.41
00370.30
24.69 %
75.31 %
total 000491.71 (J)
KE
PE
Evaluating Initialization...
gravity
earth
damping
00.0
show phase
current time
000.00


play
pause
step
reset
duration
00043.
Δt
0.020
number of bobs
1
2
3
min
m1
20.000
min
m2
15.000
min
m3
20.000
min
L1
2.00
min
L2
2.00
min
L3
2.00
zero
θ1
+000.6
zero
θ2
+000.1
zero
θ3
+000.1
zero
θ̇1
+2.30
zero
θ̇2
+0.00
zero
θ̇3
+0.00
P.E. (J)
K.E. (J)
00068.54
00211.60
24.47 %
75.53 %
total 000280.14 (J)
PE
KE
Evaluating Initialization...
gravity
earth
damping
02.4
show phase
current time
000.00


play
pause
step
reset
duration
00042.
Δt
0.030
number of bobs
1
2
3
min
m1
20.000
m2
15.000
m3
20.000
L1
2.00
L2
2.00
L3
2.00
zero
θ1
+002.5
θ2
+000.1
θ3
+000.1
+2.30
+0.00
+0.00
P.E. (J)
K.E. (J)
00702.46
00211.60
76.85 %
23.15 %
total 000914.06 (J)
PE
KE
θ (rad/sec)
(i.c)
θ
(rad)


play
pause
step
reset
duration
00029.
Δt
0.050
number of bobs 1 2 3
min m1 20.000
min m2 15.000
min m3 20.000
min L1 2.00
min L2 2.00
min L3 2.00
zero θ1 +002.2
zero θ2 +000.1
zero θ3 +000.1
zero θ̇1 +2.30
zero θ̇2 +0.00
zero θ̇3 +0.00
P.E. (J) K.E. (J)
01074.15 00370.30
74.36 % 25.64 %
total 001444.45 (J)
PE
KE
θ̇ (rad/sec)
(i.c)
θ (rad)
6
4
2
−2
−4
5
10
15
20



Details (optional)

The equations of motion of the pendulum were derived using the Lagrangian energy method. For the n-bob pendulum, there are n-second order nonlinear differential equations and n degrees of freedom. The equations are kept in their non-linear form since `NDSolve[]` was used for the solving them. *Mathematica* was used to do the analytical derivation due to the high complexity of algebra for the $n = 3$ case.

Initial position conditions can be changed by dragging the bob using the mouse. When the mouse is clicked on the display, it will pause and then the mouse can be used to drag the bob to a new location. Logic inside the Demonstration will detect which bob to drag based on the proximity of the mouse current location to known bob positions.

Below is description of each control variable shown on the Demonstration: The top buttons Labeled 'run', 'pause', 'step' and 'reset' and self explanatory and used to control the simulation. The control labeled 'duration' is used to set the maximum simulation time. When this time is reached, the simulation will restart again from $t = 0$ automatically. The control labeled 'Δt' is used to change the time step for the display. The smaller the time step, the more accurate the simulation will be, but it will take longer to complete.

The control labeled 'number of bobs' is used to change the pendulum type to either simple, double or triple. The controls below that are used to adjust the mass of the bobs, units are in kg. All units in this simulation are in SI units. The controls below that are used to change the length of each pendulum bar.

The controls below that are used to set the initial conditions. Units are in radians for the angles and in radians per second for the angular velocities. Initial angle positions can also be set using the mouse as mentioned above, however, using the controls might provide more accurate settings if needed. For convenience, small buttons next to the control variables can be used to quickly set the value to zero.

The 'g' control is used to select the gravitational constant g. The control labeled 'show phase' is used to turn on and off the phase portrait plot. The red point in the phase portrait plot indicates the initial conditions, and the black point is the position at the end of the duration. The moving blue point is the position in phase space at the current time.

The control labeled 'c' is used to change the damping coefficient. At the bottom is a display of the energy plot, it shows the current kinetic energy (KE), potential energy (PE) and the total energy in Joules.

The angles $\theta_1, \theta_2, \theta_3$ are all measured from the vertical line. When the pendulum is hanging vertically at rest, then all angles will have zero values at this position. The angles and velocities are taken to be positive in anticlockwise direction.

Reference:

1. Dare A. Wells, *Schaum's Lagrangian Dynamics*, McGraw-Hill, 1967

Control Suggestions (optional)

- [x] Resize Images
- [ ] Rotate and Zoom in 3D
- [ ] Drag Locators
- [ ] Create and Delete Locators
- [ ] Slider Zoom
- [ ] Gamepad Controls
- [x] Automatic Animation
- [ ] Bookmark Animation

Search Terms (optional)

Pendulum

Lagrangian

Related Links (optional)

Lagrangian

Orbits Of A Triple Pendulum

Simulating A Three Armed Pendulum

Authoring Information

Contributed by: Nasser M. Abbasi

