Lesson #10: Elementary Queueing Theory

- **Definitions of Queueing Systems**
- **Queueing System Simulation**
- **Birth-Death Queueing Theory (Elementary Queueing Theory)**

An "in-progress" demonstration of queue and workload

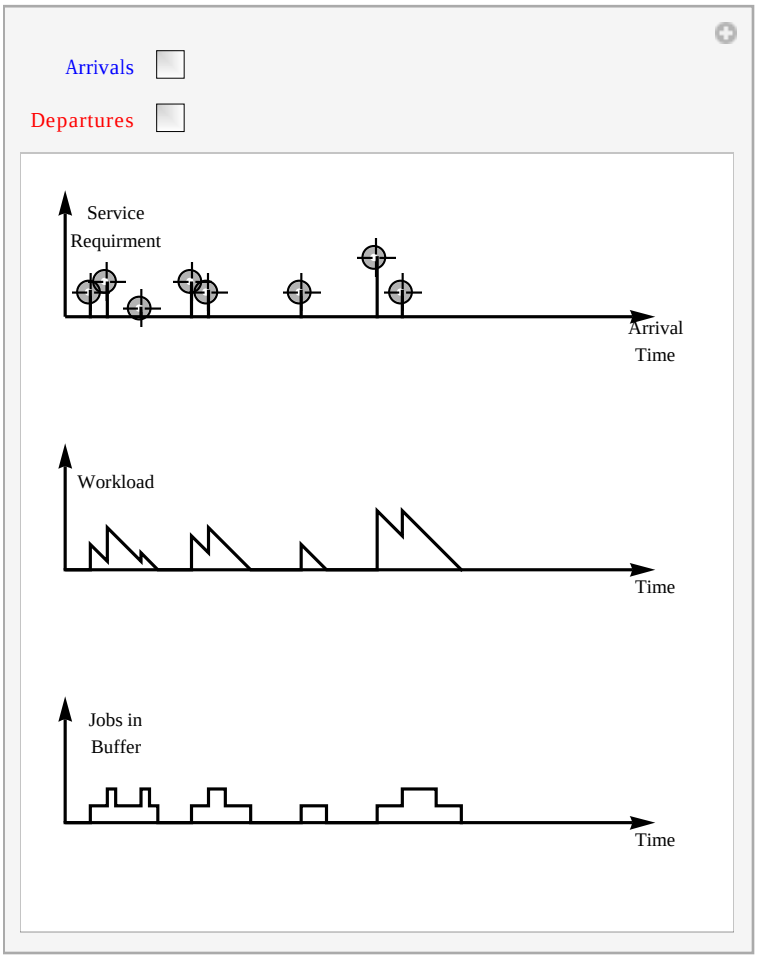
- **A function for simulating single server queues based on a marked point process input**

In[1]:=
```
queueJump = 2;
sim[mpp_List] := Module[{},
   workloadProcess = {{0.0, 0.0}};
   departureTimes = {};
   workload = 0.0; previousWorkload = 0.0;
   time = 0.0;
   mppIndex = 1;
   lastDepartureTime = Infinity;
   (*Loop to create workload and departure times*)
   While[mppIndex ≤ Length[mpp] || 0 < workload,
    arrivalTime = If[ mppIndex ≤ Length[mpp], mpp[[mppIndex]][[1]], Infinity];
    If[lastDepartureTime < arrivalTime,
     time = Last[departureTimes];
     workload = 0.0;
     AppendTo[workloadProcess, {time, workload}];
     lastDepartureTime = Infinity;
     , (*Else*)
     workload = Max[workload - (arrivalTime - time), 0];
     time = arrivalTime;
     AppendTo[workloadProcess, {time, workload}];
     workload += mpp[[mppIndex]][[2]];
     AppendTo[workloadProcess, {time, workload}];
     AppendTo[departureTimes, lastDepartureTime = time + workload];
     ++mppIndex;
    ];
   ];
   (*Generate buffer process from arrival and departure times*)
   bufferChanges = Sort[Join[Map[{#[[1]], queueJump} &, mpp],
      Map[{#, -queueJump} &, departureTimes]], #1[[1]] < #2[[1]] &];
   bufferProcess = FoldList[
     {#2[[1]], #1[[2]] + #2[[2]]} &
     , {0.0, 0}, bufferChanges];
   lastVal = Last[bufferProcess];
   bufferProcess = Transpose[{Drop[bufferProcess, -1], Rest[bufferProcess]}];
   bufferProcess = Flatten[Map[{#[[1]], {#[[2, 1]], #[[1, 2]]}} &, bufferProcess], 1];
   AppendTo[bufferProcess, lastVal];
   (*return realizations*)
   Return[{workloadProcess, departureTimes, bufferProcess}];
  ];
mppOffset = 30;
bufferOffset = -30;
timeHorizon = 70;
paddRatio = 1.3;
maxService = 10;
yaxisLength = 15;
MarkedPoint[{t_, s_}] := Line[{{t, mppOffset}, {t, Max[mppOffset + s, mppOffset]}}];
```

The demonstration

In[10]:=
```
Manipulate[
 pts = Map[{#[[1]], Max[#[[2]] - mppOffset, 0] + mppOffset} &, pts];
 mpp =
  Sort[Select[Map[{#[[1]], #[[2]] - mppOffset} &, pts], #[[2]] > 0 &], #1[[1]] < #2[[1]] &];
 simOut = sim[mpp];
 Graphics[Join[
   {Black, Dashing[{}], Thickness → 0.005,
    Translate[Line[simOut[[3]] ], {0, bufferOffset}]},
   { White, Rectangle[{0, 0}, {3 * timeHorizon, 5 * yaxisLength}],
    Black, Dashing[{}], Thickness → 0.005, Line[simOut[[1]]]},
   {White, Rectangle[{0, mppOffset}, {3 * timeHorizon, mppOffset + 5 * yaxisLength}],
    Black, Dashing[{}], Thickness → 0.005}, Map[MarkedPoint, mpp],
   {Text["Arrival\nTime", {timeHorizon, mppOffset - 3}],
    Text["Time", {timeHorizon, 0 - 2}],
    Text["Time", {timeHorizon, bufferOffset - 2}],
    Text["Service\nRequirment", {6, mppOffset + yaxisLength * 0.7}],
    Text["Workload", {6, 0 + yaxisLength * 0.7}],
    Text["Jobs in\nBuffer", {6, bufferOffset + yaxisLength * 0.7}]},
   {Arrow[{{0, 0}, {0, yaxisLength}}],
    Arrow[{{0, mppOffset}, {0, mppOffset + yaxisLength}}],
    Arrow[{{0, bufferOffset}, {0, bufferOffset + yaxisLength}}],
    Arrow[{{0, mppOffset}, {timeHorizon, mppOffset}}],
    Arrow[{{0, bufferOffset}, {timeHorizon, bufferOffset}}],
    Arrow[{{0, 0}, {timeHorizon, 0}}]},
   {Dashing[{Tiny}]},
   If[arrivalLinesQ, Join[
     {Blue},
     Map[Line[{{#[[1]], mppOffset}, {#[[1]], bufferOffset}}] &, mpp],
     {PointSize → Medium, Point[Map[{#[[1]], 0} &, mpp]]}], {}],
   If[departureLinesQ, Join[
     {Red},
     Map[Line[{{#, 0}, {#, bufferOffset}}] &, simOut[[2]]],
     {PointSize → Medium, Point[Map[{#, 0} &, simOut[[2]]]]}], {}]
  ],
  PlotRange → {{-1, timeHorizon * 1.07}, {bufferOffset * paddRatio, mppOffset + yaxisLength}},
  Axes → False, AxesOrigin → {0, 0}],
 {{arrivalLinesQ, False, Style["Arrivals", Blue]}, {True, False}},
 {{departureLinesQ, False, Style["Departures", Red]}, {True, False}},
 {{pts, {{3, 3 + mppOffset}, {5, 4 + mppOffset}, {9, 1 + mppOffset},
    {15, 4 + mppOffset}, {17, 3 + mppOffset}, {28, 3 + mppOffset},
    {37, 7 + mppOffset}, {40, 3 + mppOffset}}}, Locator, LocatorAutoCreate → True,
  SaveDefinitions → True}]
```

Out[10]=



Also look at the demo (and source code in Wolfram Demo Project): Scalings of a GI/G/1 Queue Realization