the resource to that color. For example, **Figure 6.13** uses this construction to randomly produce red and blue resources with an average proportion of 7/2.

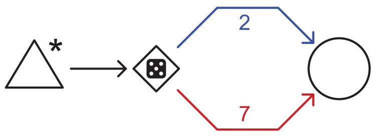


FIGURE 6.13
Producing resources with random colors

Delays and queues can use color coding. By giving them outputs with different colors, they will delay resources of the corresponding color by a number of time steps indicated by that output. For example, **Figure 6.14** represents the mechanics of a game where players can build knights and soldiers. Knights are represented as red resources, and soldiers are represented as blue resources. Knights cost more gold and take more time to build.

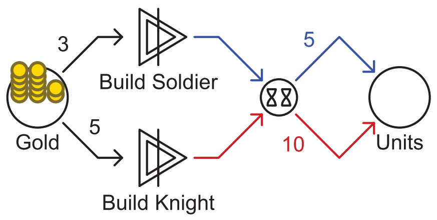


FIGURE 6.14
Using color coding to build different units with one building queue

Feedback Structures in Games

The structure of a game's internal economy plays an important role in a game's dynamic behavior and gameplay. In this structure, feedback loops play a special role. A classic example of feedback in games can be found in *Monopoly* where the money spent to buy property is returned with a profit because more property will generate more income. This feedback loop can be easily read from the Machinations diagram of *Monopoly* (**Figure 6.15**): It is formed by the closed circuit of resource and state connections between the Money and Property pools.

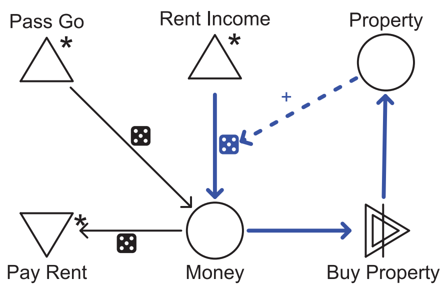


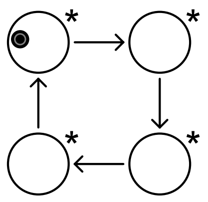
FIGURE 6.15
Monopoly

Closed Circuits Create Feedback

A feedback loop in a Machinations diagram is created by a closed circuit of connections. Remember that feedback occurs when state changes create effects that ultimately feed back to the original element. A closed circuit of connections will cause this effect.

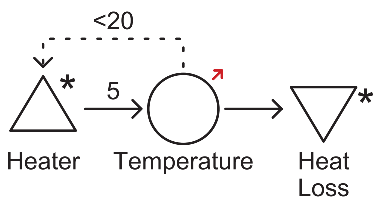
A closed circuit of only resource connections (as in **Figure 6.16**) cannot display very complex behavior. The resource is pulled through the pools in circles creating a simple periodic system, but nothing more interesting can happen.

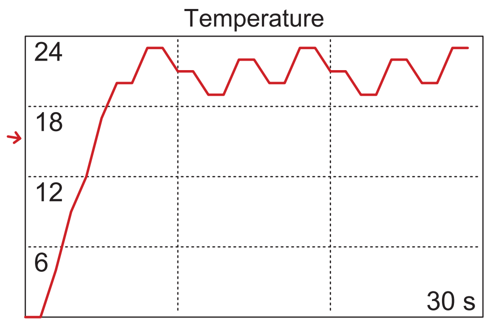
FIGURE 6.16 Feedback created by only resource connections. The resource simply goes round and round.



The most interesting feedback loops consist of a closed circuit that mixes resource connections and state connections. The loop should contain at least one label modifier or activator. For example, the mechanism in **Figure 6.17** uses an activator to maintain the resources in the pool at about 20. It acts the way a heating system does to keep a room warm in cold climates: It turns on a heater with a fixed output when the temperature drops below 20. The graph in the same figure displays the temperature over time.

FIGURE 6.17 Heater feedback mechanism using an activator





CHARTS IN MACHINATIONS DIAGRAMS

Using the online tool for Machinations diagrams, it is very easy to produce charts tracking the number of resources in a pool over time. The chart in Figure 6.17 is produced by the tool. You can add a chart to a diagram like any other element. To start measuring the number of resources, simply connect a pool to the chart with a state connection. When selected, this connection is displayed as normal, but when it is not selected, it is reduced to two small arrows to avoid visual clutter.

You can build a similar system using a label modifier instead (**Figure 6.18**). In this case, the heater's output rate is adjusted by the actual temperature, creating a more subtle temperature curve. This simulates a heater that can produce varying amounts of heat rather than a fixed amount, as in Figure 6.17.

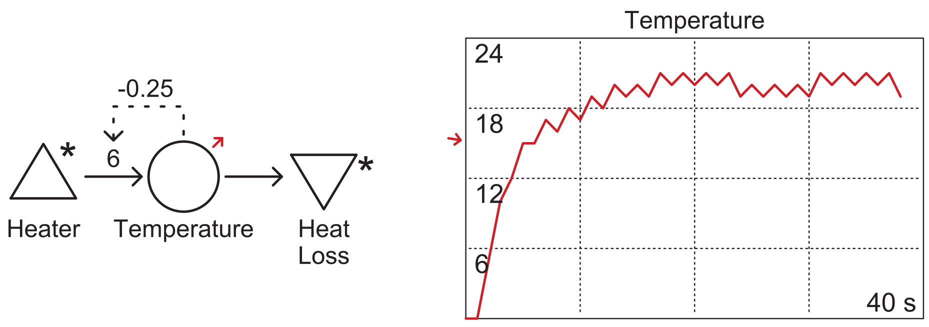


FIGURE 6.18
Heater feedback mechanism using a label modifier

Feedback by Affecting Outputs

A feedback loop can also be created by a circuit of connections that affects the output of an element. For example, consider a Machinations diagram for an air conditioner (**Figure 6.19**). The higher the temperature, the faster the temperature is drained.

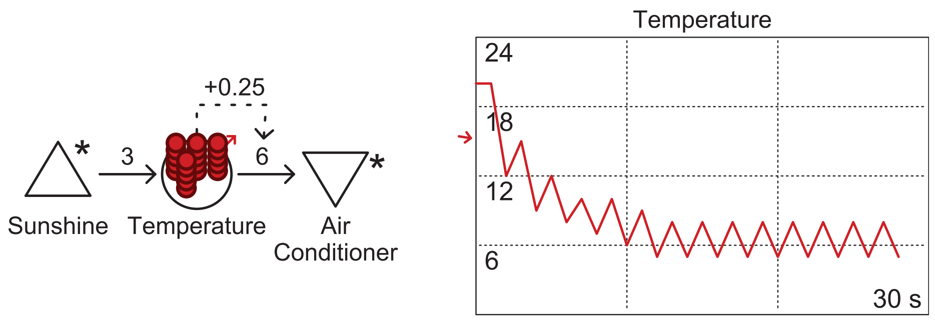


FIGURE 6.19
An air conditioner in Machinations

Any changes that affect the output of a node can also close a feedback loop. In this case, output can be affected directly by a label modifier or by a trigger or activator affecting an element at the end of the resource that is able to pull resources (such as a drain, converter, or gate, as in **Figure 6.20**).

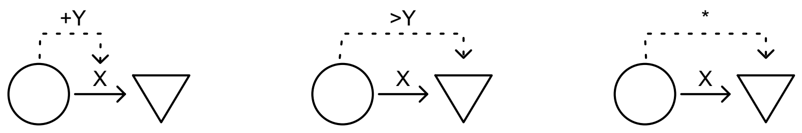


FIGURE 6.20
Closing a feedback loop by affecting the output

Positive and Negative Feedback Basketball

In Chapter 4, we briefly introduced Marc LeBlanc's concept of positive and negative feedback basketball to explain the effects of positive and negative feedback on games. In this section, we'll discuss the idea in more detail and show how to model it in Machinations.

Positive feedback basketball is played like normal basketball but with the following extra rule: "For every N points of difference in the score, the team that is ahead may have an extra player in play." **Figure 6.21** shows a model of positive feedback basketball. It uses a very simple construction to model basketball itself: Every player on a team has a particular chance to score every time step. Teams initially consist of five players, so their chance to score is represented as a source with an initial production rate of five (for simplicity, we assume that a basket is worth one point, not two, and we ignore three-point shots and free throws). Next comes a gate with a percentage; this indicates the percentage of player attempts that actually *succeed* in scoring. As you can see, the blue team is much better than the red team is; the blue team's chance of scoring is 40% while the red team's chance is only 20%. A pool called *Difference* keeps track of the difference in the points scored, because every time blue scores, a point is added to the *Difference* pool, and every time red scores, a point is subtracted. Each team can field one more player for every five points that it leads by—if it's ahead by five, it gets one more player; if it's ahead by 10, it gets two; and so on. The development of the scores and the difference in the scores are plotted over time in the chart. As you can see, the score of the better team quickly spirals upward as the difference increases, allowing them to field more and more players.

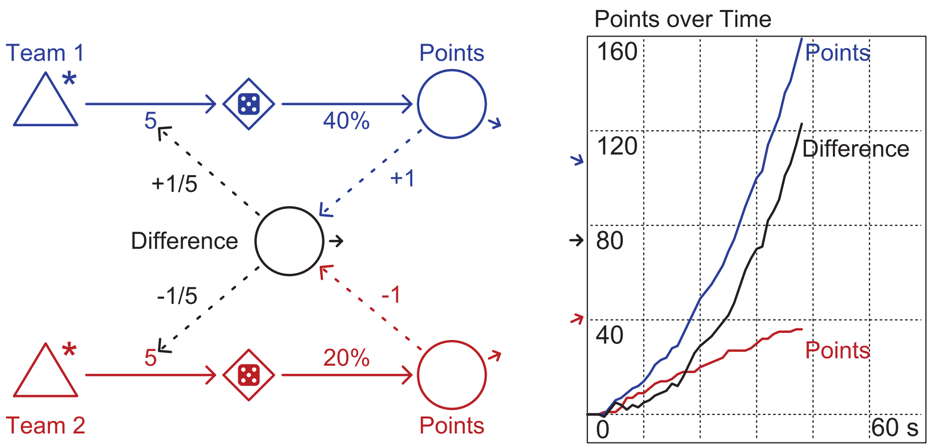


FIGURE 6.21 Positive feedback basketball. Team sizes are prevented from dropping below 5 by setting their minimal value to 5.

In negative feedback basketball, the extra rule is reversed: The losing, rather than the leading, team can field an extra player for every N points of difference. Again, this can be easily modeled as a Machinations diagram, simply by reversing the signs of a few state connections. The chart that is produced by running the diagram is harder to predict. Without looking at **Figure 6.22**, can you guess what happens to the points of both teams and the difference in scores?

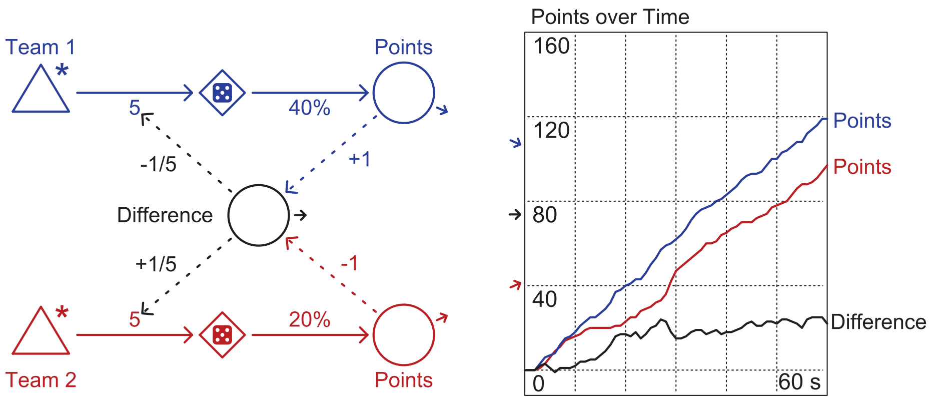


FIGURE 6.22 Negative feedback basketball

The chart in **Figure 6.22** surprised us when we first produced it. Where you might expect the negative feedback to cause the poorer team to get ahead of the better team, it never does. What happens is that the negative feedback stabilizes the difference between the two teams. At some point, the poorer team is so far behind that their lack of skill is compensated for by their team size, and beyond this point, the difference doesn't change much.

Another interesting effect of feedback occurs when you have two teams of similar skill play positive feedback basketball. In that case, both teams will score at a more or less equal rate. However, once one of the teams by chance takes a lead, the positive feedback kicks in, and they can field more and more players. The result might look something like **Figure 6.23**.

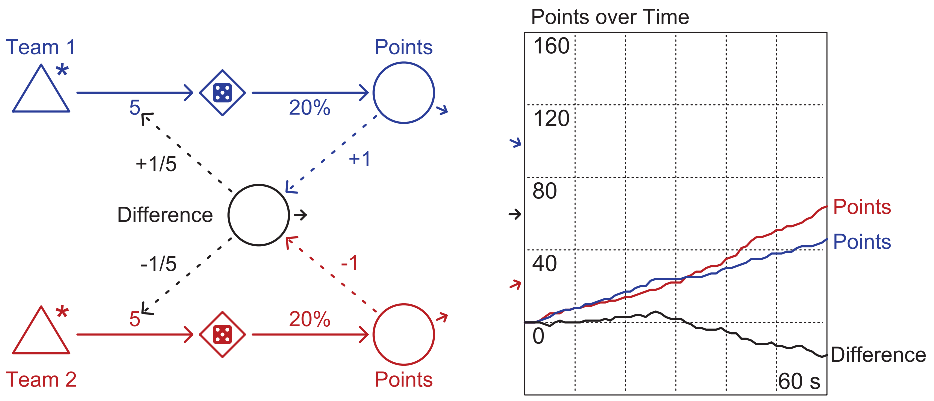


FIGURE 6.23 Positive feedback basketball between two equal teams. Notice the distinctive slope in the line that depicts the difference between the scores after roughly 30 steps.