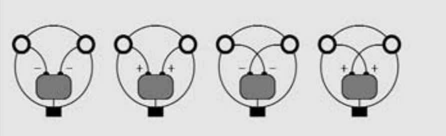
Tissue

Casey Reas

Tissue exposes the movements of autonomous software machines. Each line in the image reveals the path of each machine's movement as it responds to stimuli in its environment. People interact with the software by positioning the stimuli on the screen. Through exploring different positions of the stimuli, an understanding of the total system emerges from the subtle relations between the simple input and the resulting fluid visual output.

The concept for *Tissue* originated with the book *Vehicles, Experiments in Synthetic Psychology* by Valentino Braitenberg, a neuroanatomist. As Braitenberg spent his career counting fibers in the visual ganglia of flies and synapses in the cerebral cortex of mice, he began to distill generalizations about how nervous systems work. He states, “I have been dealing for many years with certain structures within animal brains that seemed to be interpretable as pieces of computing machinery because of their simplicity and/or regularity [Braitenberg p.1].“ He uses the term “synthetic psychology” to define the construction of personality/behavior through building up structures of “computing machinery.” In *Vehicles,* Braitenberg defines a series of 13 conceptual constructions by gradually building more complex behavior with the addition of more machinery.

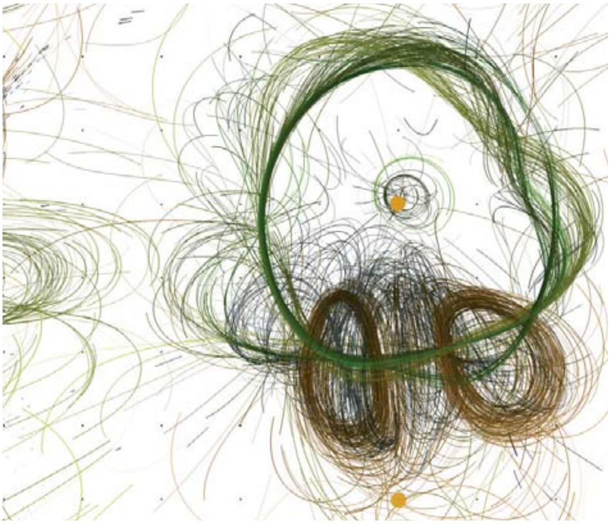
The software machines in *Tissue* are analogous to Braitenberg’s Vehicle 4. Simple layers of code combine to create the deceptively complicated behavior of these machines. Each machine has two software sensors to detect stimuli in the environment and two software actuators to propel itself. The relationships between the sensors and actuators determine the specific behavior for each machine. In the thousands of software machines simultaneously running in *Tissue,* there are four distinct types, each specified with a color. The architectures of the *Tissue* machines are shown in Figure 1. Each machine has a variable speed, direction, and position of movement and all machines share the same size, turning rate, and maximum speed. They are constrained to their environment and when they hit the edge, they reverse direction.

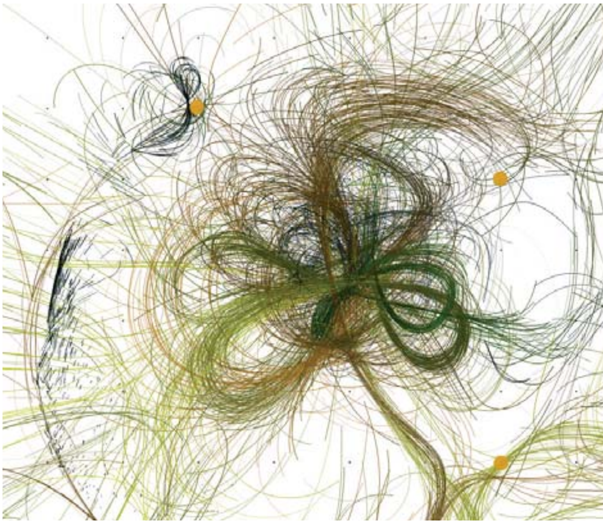


Each machine continually alters its direction and speed by analyzing its position in relation to the environment. First, it determines the distance from each of its sensors to the first stimulus point:

```
dx = stimulusX – leftSensorX;
dy = stimulusY – leftSensorY;
leftDistance = sqrt(dxdx + dydy);
dx = stimulusX – rightSensorX;
dy = stimulusY – rightSensorY;
rightDistance = sqrt(dxdx + dydy);
```

The software then normalizes the two distance measurements, “leftDistance” and “rightDistance”, and compresses the values between 0.0 and 1.0. The resulting values are





input into a nonlinear function, returned, and saved. These two new values are averaged and stored into a variable called “normSensorAverage:”

```
normSensorLeft = leftDistance/maxDistance;
normSensorRight = rightDistance/maxDistance;
normSensorLeft =  hump(normSensorLeft);
normSensorRight =  hump(normSensorRight);
normSensorAverage = (normSensorLeft + normSensorRight) / 2.0;
```

The speed of the machine is then modified using the value of “normSensorAverage” and the constant value “maxSpeed,” which is the maximum speed that a machine is capable of reaching:

```
speed = speed + (maxSpeed – (maxSpeed * normSensorAverage));
```

As mentioned previously, the software has four distinct types of machines. The two lines of code below modify the direction for one type of machine in relation to its current direction, turning speed, and the distance from each sensor to the stimulus:

```
direction = direction + leftDistance/turnSpeed * (1-normSensorLeft);
direction = direction – rightDistance/turnSpeed * (1-normSensorRight);
```

The same calculations are made for each additional stimulus and then the X and Y positions of the machine are updated based on the new speed and position. These calculations occur for each of the thousands of machines that are simultaneously displayed on the screen.

People interact with *Tissue* by positioning stimuli in the environment. They experience an interesting balance of control as their actions have a direct, but imprecise effect on the software. Small changes in the positions of the stimuli create large global changes in the movements of the machines. It is not possible to dominate the software machines, but their behavior can be encouraged and intuitively understood. *Tissue* is an example of software creating a fluid and unexpected interaction rather than more typical rigid and mathematical constructions.

Additional information, images, video clips, and source code for *Tissue* is available at *http://www.groupc.net*

Braitenberg, Valentino. *Vehicles, Experiments in Synthetic Psychology.* MIT Press, Boston 1984.