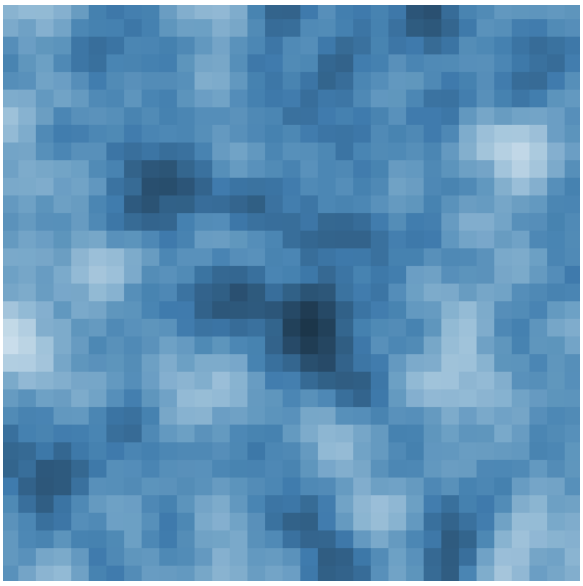
REPAST MODELS LIBRARY: DIFFUSION (RELOGO)

TIM SWEDA



Introduction

This model demonstrates the implementation of @Diffusible variables for patches, which are useful for representing quantities that tend to become less localized over time. Some examples include heat, density of a gas, and concentration. ReLogo provides several methods exclusively for @Diffusible patch variables that make it easy to manipulate their values.

In this Diffusion model, an initial pattern of colored patches is allowed to diffuse until the color of the entire grid is uniform. The user can specify the starting pattern as well as the rate of diffusion, and observe how diffusion occurs under those settings.

Usage

After selecting a **Color** for coloring patches and an initial **Pattern** from the drop-down menus, click the **Setup** button to initialize and the **Go** button to run the model. The choice of **Color** is purely aesthetic and has no effect on how the model runs. The choices for **Pattern** determine which patches are colored initially and are as follows:

- **Random100**: 100 random patches
- **Random300**: 300 random patches
- **Random500**: 500 random patches
- **OppositeEdges**: left and right edges (width = 4)
- **AdjacentEdges**: left and top edges (width = 4)
- **Spot**: 7x7 square in the center
- **Grid**: patches in every third row and column

If the color becomes too faint after diffusing some, the **Darken** button can be clicked to make all of the patches 10% darker. The **Diffusion Rate** slider can also be adjusted to modify the rate at which the color diffuses from the patches. The value of the slider equals the fraction of the patch variable that is diffused to neighboring patches during one time step.

The values **maxConcentration** and **minConcentration** measure the maximum and minimum values, respectively, of the color concentration in any one patch. When these values are within 0.001 of each other, the simulation stops since the diffusion process is effectively complete. At this point, all patches should appear to be the same color.

Exercises

- Experiment with the different starting patterns to get a feel for which ones diffuse quickly and which ones take longer to diffuse completely. Try to come up with some simple metrics for qualitatively predicting how long a pattern will take to diffuse.
- Reprogram the way that patches are colored so that patches with **concentration** > 0.9 are red, patches with **concentration** < 0.5 are green, and the remaining patches are yellow. Retain the shading feature so that you can still watch diffusion occurring.
- Add some turtles to the model to execute some behavior that makes the diffusion more interesting.

ReLogo Features

The Diffusion model showcases two different methods for modifying @Diffusible patch variables: **diffuse** and **diffusibleMultiply**. The **diffuse** method takes as inputs the name of the patch variable as a String and a double value between 0 and 1 representing the fraction of the patch variable to be shared with neighboring patches. The amount to be shared is split equally among all neighbors. For the **diffusibleMultiply** method, the first input is also a String of the patch variable's name, but the second input is a double by which the @Diffusible variable of each patch is multiplied. Similar methods exist for dividing, adding to, and subtracting from patch variables: **diffusibleDivide**, **diffusibleAdd**, and **diffusibleSubtract**. There is also a **diffusibleApply** method, where the second input is a DoubleFunction that can be applied to the patch variable.

In addition, the model demonstrates how to change not only the color of patches, but also their shade. There are a few simple rules to keep in mind when working with colors in ReLogo:

- Each double value between 0.0 and 139.9, inclusive, with one digit to the right of the decimal point has a corresponding color.
- If x is a double value, then x has the same color as x mod 140 with any digits to the right of the first after the decimal point truncated (e.g., 147.099 and 7.0 both correspond to the same color).
- Every value ending in 5.0 has a name (e.g., **red()**, **blue()**, etc.).
- Values ending in 0.0 appear black; **black()** = 0.0.
- Values ending in 9.9 appear white; **white()** = 9.9.
- The tens and hundreds digits determine the color; the last two digits determine the shade, with higher values corresponding to lighter shades.