Chapter 1

Working with Patches

Create a NetLogo model file called *Class2.nlogo*, put your work in it, and submit it for today's in-class exercises assignment.

1.1 Basic

1. Write a command procedure called `basic1` that when executed calls `clear-all` and then sets the color of all patches whose x-coordinate equals 6 or more to yellow.

   Here's a stub for you to work with:

   ```
   to basic1
     clear-all
     [** Your code here. **]
   end
   ```

   Answer:

   ```
   to basic1
     clear-all
     ask patches with [pxcor > 5] [set pcolor yellow]
   end
   ```

   ---

2. Write a command procedure called `basic2` that when executed calls `clear-all` and then colors each patch yellow with probability 12/100, and then for each patch if it has 3 or more yellow neighbors, sets the patch color to red and sets it label to `***`

Here's a stub for you to work with:

```
to basic2
  clear-all
  [** Your code here. **]
end
```

Answer:

```
to basic2
  clear-all
  ask patches [if random 100 < 12
    [set pcolor yellow]]
  ask patches [if count neighbors with [pcolor = yellow] >= 3
    [set plabel "***"
      set pcolor red]
  ]
end
```

---

3. Make a new slider on the interface with range from 0 to 100 in increments of 1 which is called spawn-percentage. Change the setup method so that, on average, spawn-percent percent of the patches begin alive at setup.

   You should modify the current setup procedure.

Answer:

```
to setup
  ca
  ask patches [
    set pcolor blue - 3
    if random 100 < spawn-percentage [ set pcolor green ]
  ]
  reset-ticks
end
```

---

4. Write a new setup procedure called make-glider that clears the world and makes a glider with its top-left corner at (0,0) which moves down and to the right (see pg. 64).

   Here's a stub for you to work with:

```
to make-glider
  clear-all
  [** Your code here. **]
end
```

Answer:

```
to make-glider
  ca
  ask patches [
    set pcolor blue - 3]
    ask patch 0 0 [set pcolor green]
    ask patch 1 -1 [set pcolor green]
    ask patch 2 -1 [set pcolor green]
    ask patch 0 -2 [set pcolor green]
    ask patch 1 -2 [set pcolor green]
  ]
  reset-ticks
end
```

---

5. Find a still-life configuration with five live cells and write a new procedure called make-still which clears the world and creates the still configuration at (0, 0).

   Here's a stub for you to work with:

```
to make-still
  clear-all
  [** Your code here. **]
end
```

Answer:

```
to make-still
  ca
  ask patches [
    set pcolor blue - 3]
    ask patch 0 0 [set pcolor green]
    ask patch 0 -1 [set pcolor green]
    ask patch 1 0 [set pcolor green]
```

```
    ask patch 2 -1 [set pcolor green]
    ask patch 1 -2 [set pcolor green]
end
```

---

6. Find a period two oscillator different from the blinker and write a new procedure call make-oscillator which creates it at the origin.

   Here's a stub for you to work with:

```
to make-oscillator
  clear-all
  [** Your code here. **]
end
```

Answer:

```
to make-oscillator
  ca ask patches [
    set pcolor blue - 3
    ask patch 0 0 [set pcolor green]
    ask patch 1 0 [set pcolor green]
    ask patch 2 0 [set pcolor green]
    ask patch 1 -1 [set pcolor green]
    ask patch 2 -1 [set pcolor green]
    ask patch 3 -1 [set pcolor green]
  ]
  reset-ticks
end
```

1.2 Beyond the Basics

1. Write a procedure called go2 which implements an altered set of transition rules. If a patch is set to die in the old rules, instead of changing its pcolor to blue, change it to yellow. All yellow patches should die at the beginning of each tick.

   Here is the basic go procedure for you to work with:

```
to go
  ask patches [
```

```
    set live-neighbors count neighbors with [pcolor = green]
  ]
  ask patches [
    if live-neighbors = 3 [set pcolor green]
    if live-neighbors = 0 or live-neighbors = 1 [set pcolor blue - 3]
    if live-neighbors >= 4 [set pcolor blue - 3]
  ]
end
```

Answer:

```
to go2
  ask patches [
    set live-neighbors count neighbors with [pcolor = green]
  ]
  ask patches [
    if pcolor = yellow [set pcolor blue - 3]
    if live-neighbors = 3 [set pcolor green]
    if (live-neighbors = 0 or live-neighbors = 1) and pcolor = yellow [set
    if (live-neighbors = 0 or live-neighbors = 1) and pcolor = green [set
    if live-neighbors >= 4 and pcolor = yellow [set pcolor blue - 3]
    if live-neighbors >= 4 and pcolor = green [set pcolor yellow]
  ]
end
```

---

2. Create four sliders, min-spawn, max-spawn, min-die, and max-die, each of which scales between 0 and 8. Write a new procedure called go3 on the model of the basic one, except replacing the hard-coded numbers with slider values.

   Here is the basic go procedure for you to work with:

```
to go
  ask patches [
    set live-neighbors count neighbors with [pcolor = green]
  ]
  ask patches [
    if live-neighbors = 3 [set pcolor green]
    if live-neighbors = 0 or live-neighbors = 1 [set pcolor blue - 3]
    if live-neighbors >= 4 [set pcolor blue - 3]
```

```
  ]
end
```

Answer:

```
to go3
  ask patches [
    set live-neighbors count neighbors with [pcolor = green]
  ]
  ask patches [
    if (live-neighbors >= min-spawn and
        live-neighbors <= max-spawn) [set pcolor green]
    if live-neighbors <= max-die [set pcolor blue - 3]
    if live-neighbors >= min-die [set pcolor blue - 3]
  ]
end
```
