EECS 1012: Introduction to Computer Science

November 25, 2016

Rest of the course

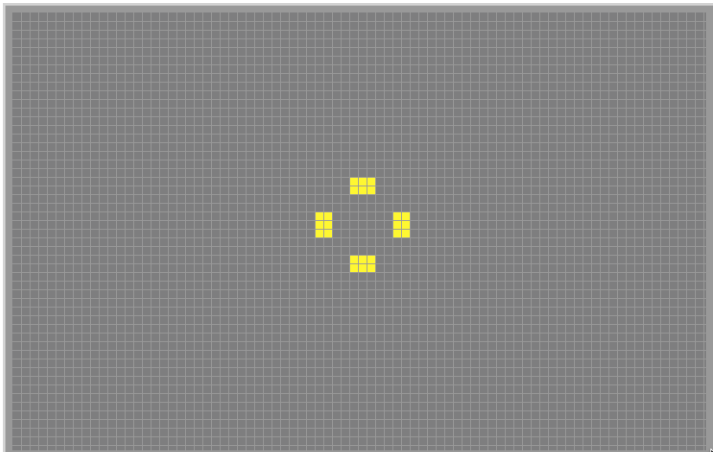
- Lectures
  - This week - advanced topics
  - Next week - advanced topics, review
  - December 5th - final quiz in class
- Labs
  - This week eCommerce
  - Next week LabTest 2

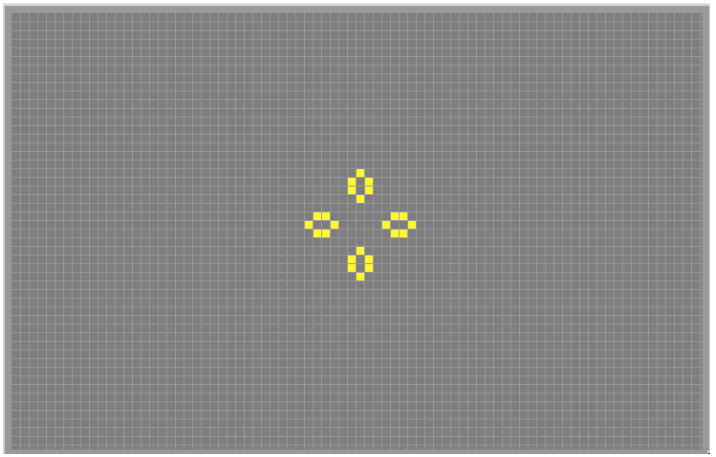
Today's example

- Conway's "Game of Life" (Scientific American, 1970)
- Goal was to develop a machine that could make copies of itself (von Neumann 1940's).

Game of life

- Played on a 2D grid of cells, each cell is either alive or dead.
- Game is played in rounds. In each round
  - A live cell with fewer than 2 or more that 3 neighbours dies, otherwise lives to next generation.
  - A dead cell with 3 neighbours becomes live in the next generation





Basic computational problem

- For the cell marked '*' need to count the number of adjacent cells that are either alive (full) or dead (empty)

A	B	C
D	*	E
F	G	H

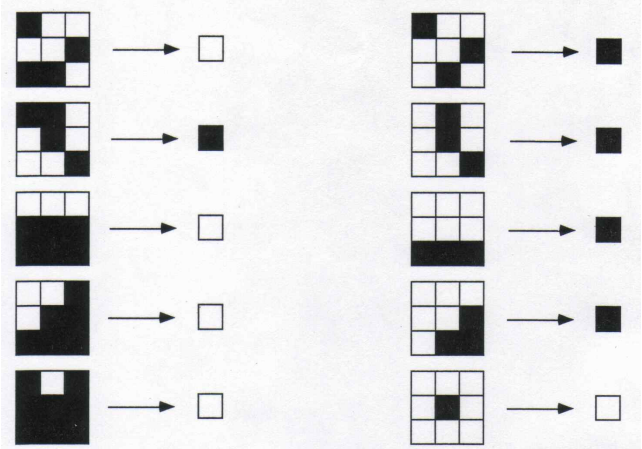
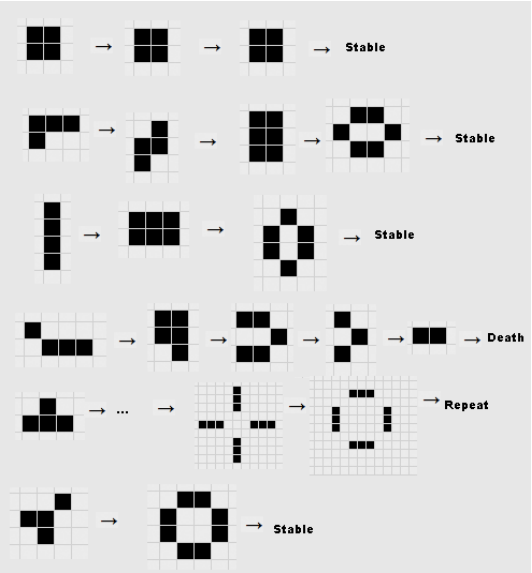


Figure 1.5.7 The CA rule Conway's Game of Life is illustrated for a few cases. When there are fewer than three or more than four neighbors in the 3 × 3 region the central cell is OFF in the next step. When there are three neighbors the central cell is ON in the next step. When there are four neighbors the central cell retains its current value in the next step. This rule was designed to capture some ideas about biological organism reproduction and death where too few organisms would lead to disappearance because of lack of reproduction and too many would lead to overpopulation and death due to exhaustion of resources. The rule is simulated in Fig. 1.5.8 and 1.5.9. ∎



Representing 1D things in JavaScript

- So to represent 1D things (vectors, lists, etc.) in JavaScript, typically use an Array
  - var x = new Array(10);
  - x.length == 10
  - x[0], … x[1] are the values

Representing 2D thing in JavaScript

- Make each array hold … an array
- var x = new Array(3);
- x[0] = new Array(2);
- x[1] = new Array(2);
- x[2] = new Array(2);
- x[0][0], x[0][1], x[1][0], x[1][1], … x[2][1]
- x.length, x[0].length, x[1].length

So making an empty world

```
function createEmptyWorld(w, h) {
  "use strict";
  var world = new Array(h);
  for(var i=0;i<h;i++) {
    world[i] = new Array(w);
    for(var j=0;j<w;j++) {
      world[i][j] = EMPTY;
    }
  }
  return world;
}
```

Using the world

- var z = emptyWorld(10, 10); // 10x10 world
- z[row][col] is either EMPTY or FULL
  - row in range 0..9
  - col in range 0..9

Displaying the world

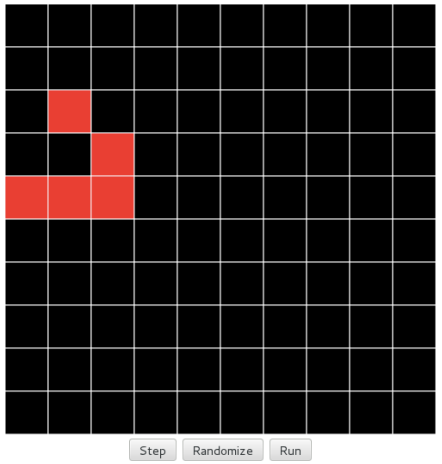
- So in HTML/JavaScript could use
  - Collection of basic elements (div's, button's, etc.)
  - Canvas

Displaying the world

```
function drawCanvas(id, world) {
  "use strict";
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");
  var w = world[0].length;
  var h = world.length;
  var cellw = Math.floor(canvas.width / w);
  var cellh = Math.floor(canvas.height / h);

  for(var i=0;i<h;i++) {
    for(var j=0;j<w;j++) {
      if(world[i][j] == EMPTY) {
        ctx.fillStyle = "#000000";
      } else {
        ctx.fillStyle = "#ff0000";
      }
      ctx.fillRect(j*cellw, i*cellh, cellw, cellh);
      ctx.strokeStyle = "#ffffff";
      ctx.strokeRect(j*cellw, i*cellh, cellw, cellh);
    }
  }
}
```

Displaying the world



“Game” logic

- Game takes place in rounds
- In each round, need to count neighbours of each cell
  - What to do at boundaries? Here will will assume the outside of the boundaries is empty

“Game” logic

```
function countNeighbours(world, w, h, i, j) {
  "use strict";
  var count = 0;
  for(var oi= -1; oi <= 1; oi++) {
    for(var oj= -1; oj <= 1; oj++) {
      if((oi!=0)||(oj!=0)) {
        var ni = i + oi;
        var nj = j + oj;
        if((ni >= 0)&&(nj >=0)&&(ni<h)&&(nj<w)&&(world[ni][nj]==FULL)) {
          count++;
        }
      }
    }
  }
  return count;
}
```

```
function applyRules(world) {
  "use strict";
  var w = world[0].length;
  var h = world.length;
  var newWorld = createEmptyWorld(w, h);
  for(var i=0;i<h;i++) {
    for(var j=0;j<w;j++) {
      var count = countNeighbours(world, w, h, i, j);
      if(world[i][j] == FULL) {
        if((count < 2)||(count > 3)) {
          newWorld[i][j] = EMPTY;
        } else {
          newWorld[i][j] = FULL;
        }
      } else {
        if(count == 3) {
          newWorld[i][j] =  FULL;
        } else {
          newWorld[i][j] = EMPTY;
        }
      }
    }
  }
  return newWorld;
}
```

One Step

```
function oneStep() {
  "use strict";
  if(!running) {
    var tmp = applyRules(globalWorld);
    globalWorld = tmp;
    drawCanvas('canvas', globalWorld);
  }
}
```

Initialization

```
function initWorld() {
  "use strict";
  var setup = [
       "           ",
       "           ",
       " X         ",
       "  X        ",
       "XXX        ",
       "           ",
       "           ",
       "           ",
       "           ",
       "           ",
       "           "
  ];
  var w = setup[0].length;
  var h = setup.length;
  var world = createEmptyWorld(w, h);
  for(var i=0;i<h;i++) {
    for(var j=0;j<w;j++) {
      if(setup[i].charAt(j) == ' ') {
        world[i][j] = EMPTY;
      } else {
        world[i][j] = FULL;
      }
    }
  }
  return world;
}
```

Looping

```
function step() {
  "use strict";
  var tmp = applyRules(globalWorld);
  globalWorld = tmp;
  drawCanvas('canvas', globalWorld);
  timer = setTimeout(step, 500);
}


function go() {
  "use strict";
  var button = document.getElementById("go");
  if(running) {
    running = false;
    button.innerHTML = "Run";
    clearTimeout(timer);
  } else {
    running = true;
    button.innerHTML = "Stop";
    timer = setTimeout(step, 500);
  }
}
```

Looping

- Note the need to clear the timeout
  - Imagine what might happen if this was not done.

Conway's Game of Life

- Fascinating mathematical exploration of self-reproducing machines.
- Provides an exploration of 2D arrays in JavaScript
  - Arrays of Arrays [of Arrays….]
- Use of canvas to display complex things (here the game board)

As an aside

- JavaScript provides ‘array of arrays’. Other languages provides 2D arrays, or both.
- Arrays of arrays - not all arrays need be the same size (different shapes are possible)

Summary

- More sophisticated programs require more thought before writing the code.
- Here we have implemented something reasonably complex, requiring us to choose the right data structures (arrays), and make a substantive number of decisions in terms of how to structure the code.