Cellular automata - making a model

Modeling sound wave

Making a simulation of given phenomenon/system begins with defining its model. One should remember that *all models are wrong* - i.e. models do not represent exactly the real world system. Model is a simplification of real system, that should focus on its most important aspects.

Spreading of sound wave is a complex process, that is dependent on different factors like: pressure, medium density, wave frequency, etc. However it can be successfully simulated as a dependency between two parameters, that describes the state of the cell:

- particle velocity,
- acoustic pressure.

Acoustic pressure is a local deviation from average atmospheric pressure, while **particle velocity** is the velocity of a real or imagined particle in a medium as it transmits a wave.

Preparations

- Dowlnoad and extract file
- Import and run project.

Sound wave simulation (2pkt)

Fill missing fragments of code in class **Board** and **Point**.

- In class **Point** create 4 variables to store its north, south, east and west neighbor.
- In class **Board** in method **initialize()**, initialize neighbors for each cell - use von Neuman neighborhood. Do not initialize the border cells.
- In class **Point** create 4 variables that store particle velocity V_a according to neighbor a, while $a \in (N, W, S, E)$
- In method **clear()** write code that reset values of particle velocity and acoustic pressure
- Implement method **updateVelocity()** according to equitation:

$$V_a(x, t+1) = V_a(x, t) - (P(x + dx_a, t) - P(x, t))$$

while $P(x, t)$ to acoustic pressure, and $V_a(x, t)$ is particle velocity in direction a, at the moment t, for cell at position defined by two values of vector x.

- Implement method **updatePresure()** according to equitation:

$$P(x, t+1) = P(x, t) - c^2 \sum_a V_a(x, t+1)$$

maximal wave velocity is defined as c, assume $c^2 = \frac{1}{2}$

- Run and test your simulation.

Model extension by non-homogenious cellular automata

Walls(2 pkt)

- In class **Point** add static array:

```
pole public static Integer []types ={0,1,2};
```

and variable

```
 int type;
```

You will use them to differentiate types of cells: air, wall, sound source (e.g. speakers)

- In constructor of class **Point** set its default type to 0 (air).
- In classes **Board** and **GUI** uncomment fragments of code that create drop-down list, which allows to choose and draw different cell types on the lattice.
- In order to implement walls, one have prevent methods **updatePresure()** and **updateVelocity()** in cells of type different than 0 (air).

Sources of sound (2 pkt)

Source emits sound wave by changes of pressure in cell. In this exercise is recommended to use sinusoidal changes of pressure.

- Add variable sinInput (type int) to class **Point**
- In class **Point** in method **updatePressure()** if cell is type 2, its pressure should change according to following conditions:

```
double radians = Math.toRadians(sinInput);
pressure = (float) (Math.sin(radians));
```

Remember to change value of variable sinInput in each step of simulation.

How one can regulate the amplitude and frequency of emitted sound ?