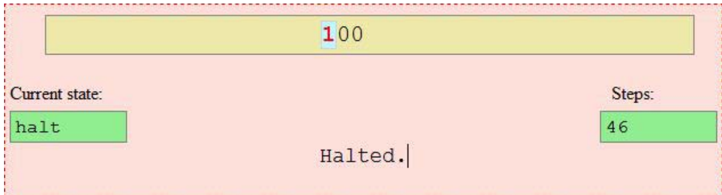
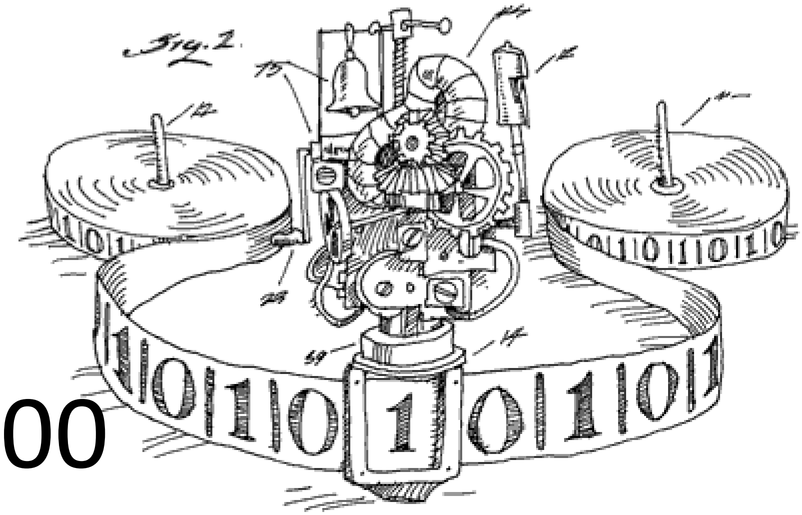
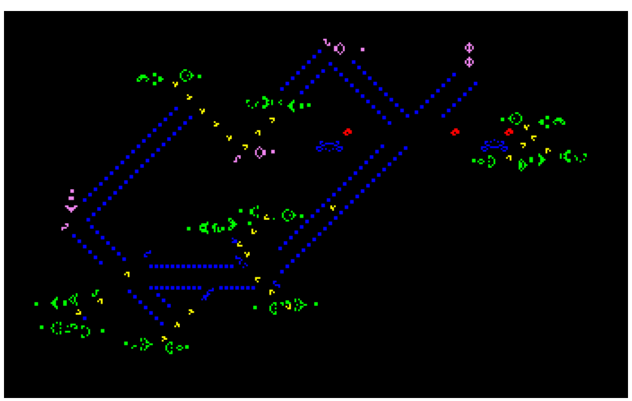




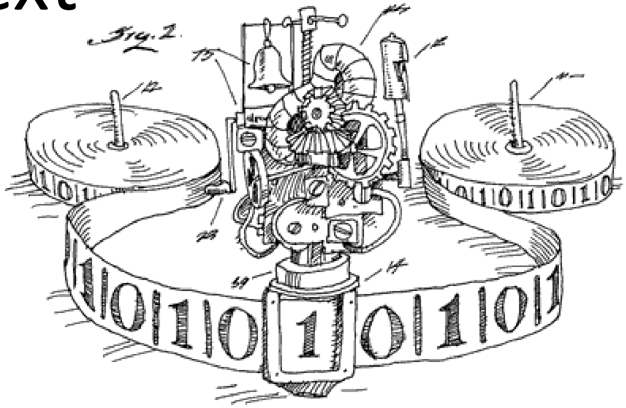
COMP2000
Logic and Computation
Lecture 2

Life vs. machine

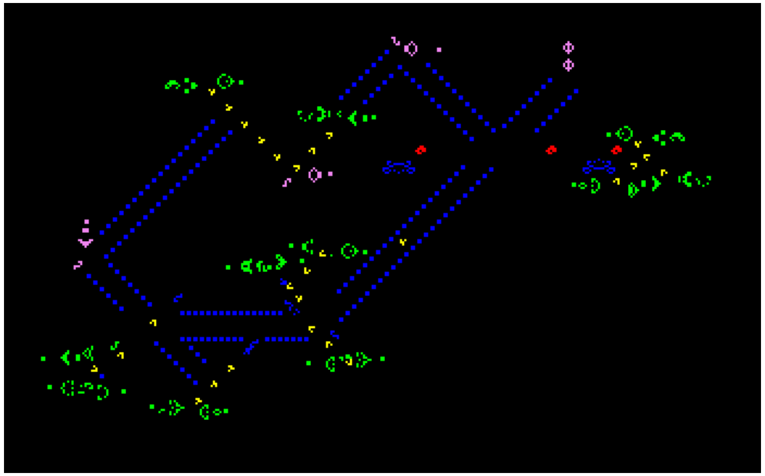


Administrative stuff

- Lab is February 15th (next Wednesday).

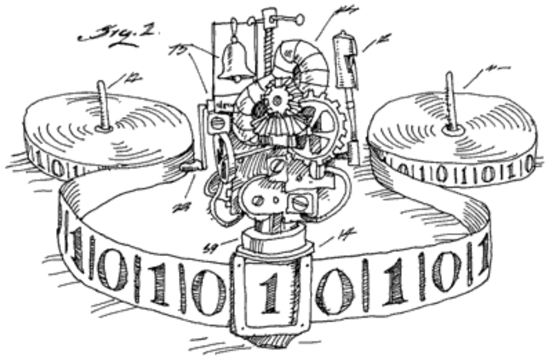


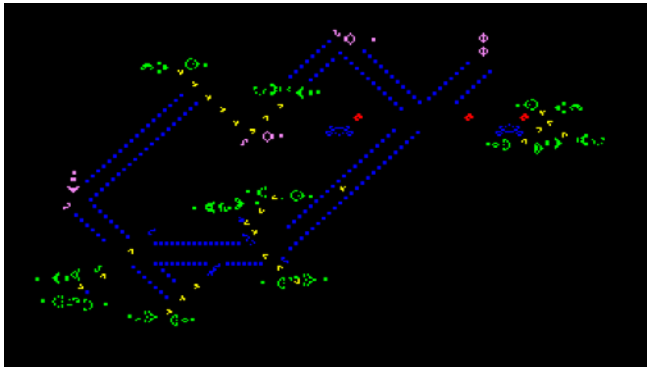
- Readings so far:
  - Chapter 1: Information
  - Chapter 2: Computation
  - Chapter 10: Cellular automata



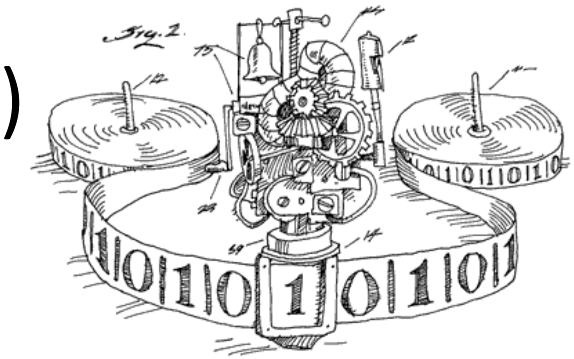
Models of computation

- In this lecture, we will talk about two (surprisingly, equivalent) models of computation
- The first one is the Turing machine
  - Our modern-day computers are based on this model
- The second is the Game of Life
  - Looks nothing like a computer, and yet has the same power.

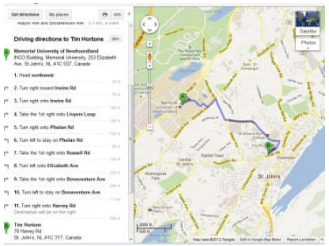




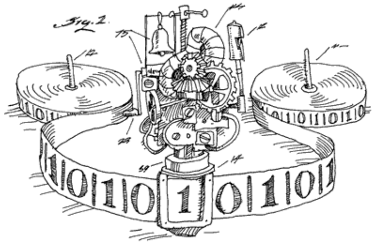
Turing machine



- A Turing machine has an (unlimited) memory, visualized as a tape
- Or a stack of paper
- And takes very simple instructions:
  - Read a symbol
  - Write a symbol
  - Move one step left or right on the tape
  - Change internal state.



Executing instructions



- Drive straight until you see the Basilica

  Internal state: looking for Basilica

  Go straight. Check for Basilica. Repeat.

- Then turn right, and drive till the next light.

  Turn right.

  Change state to “Look for traffic light”

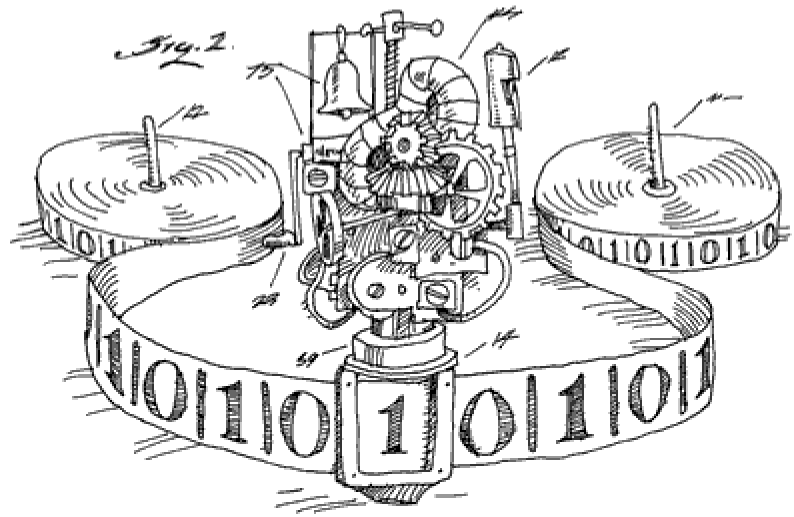
  Go straight. Check for traffic light. Repeat.

- Then turn right, and enter Tim Hortons parking lot.

  Change state to “Look for Tim Hortons”

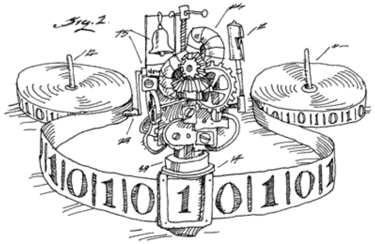
  When see Tim Hortons, turn right into the parking lot

Church-Turing thesis

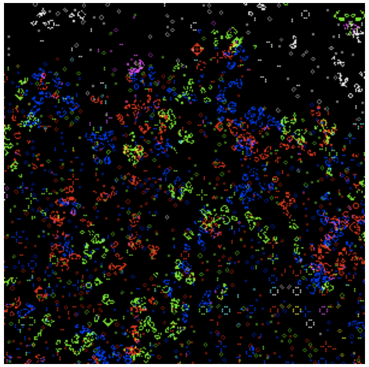


- Everything we can call "computable" in any sense of this word is computable by a Turing machine.

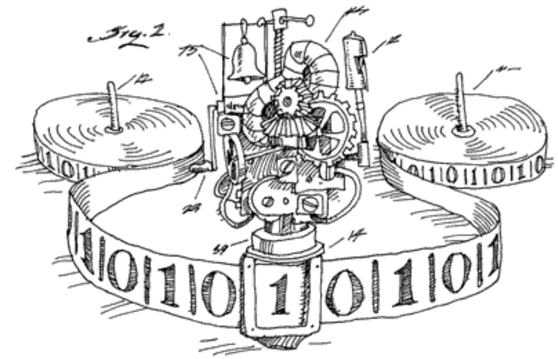
Church-Turing thesis



***Everything we can call "computable" in any sense of this word is computable by a Turing machine*.**

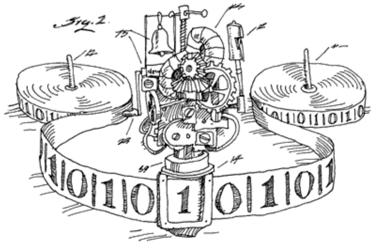


Turing machine



- A Turing machine computation starts with the tape blank except for the input
- It starts in the special start state looking at the start of the input
- Then keeps reading, writing and changing states according to the rules
- It may never stop
- If it stop, what is written on the tape is its output.

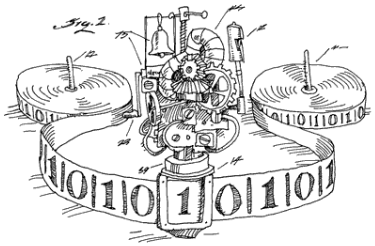
Turing machine example



- Check if the tape is empty:
  – At the start, read the first symbol
  – If it is blank, say "yes"
  – Otherwise, say "no"
- Instructions:

Current state	Reads	Writes	Moves	New state
Start_state	_	Y	Right	Halt
Start_state	0	N	Right	Halt
Start_state	1	N	Right	Halt

Turing machine example



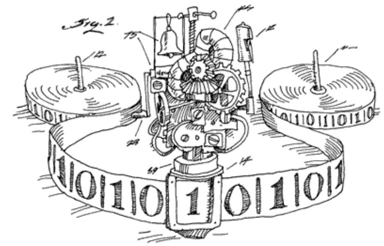
- "Check if the tape is empty" instructions:

Current state	Reads	Writes	Moves	New state
Start_state	_	Y	Right	Halt
Start_state	0	N	Right	Halt
Start_state	1	N	Right	Halt

- Simulator program:

```
0                 *   *   *   start_state        ; rename start state

start_state   _   Y   r   halt    ; if empty, write Y and stop
start_state   0   N   r   halt    ; if tape has 0, write N and stop
start_state   1   N   r   halt    ; if tape has 1, write N and stop
```

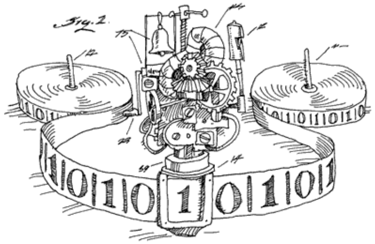
Turing machine example



- Check if the tape contains a 1:
  - At the start, read the first symbol
  - If it is 1, say “yes”
  - Otherwise, move right and repeat (keep looking)
  - Seeing a blank, say “no”
- Instructions:

Current state	Reads	Writes	Moves	New state
Start_state	_	N	Right	Halt
Start_state	0	0	Right	Start_state
Start_state	1	Y	Right	Halt

Turing machine example



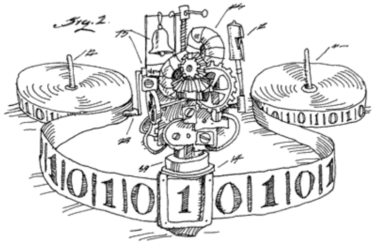
- “Check if the tape contains a 1 ” instructions:

Current state	Reads	Writes	Moves	New state
Start_state	_	N	Right	Halt
Start_state	0	0	Right	Start_state
Start_state	1	Y	Right	Halt

- Simulator program:

```
0               *   *   *   start_state    ; rename start state

start_state     _   N   r   halt           ; if reached blank, write N and stop
start_state     0   0   r   start_state    ; if still on input and no 1, repeat
start_state     1   Y   r   halt           ; if seeing a 1, write Y and stop
```

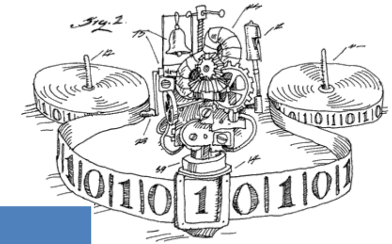
Turing machine example



- Check if the tape is empty:
  - At the start, read the first symbol
  - If it is blank, say “yes”
  - Otherwise, erase the tape and say “no”
- Instructions:

Current state	Reads	Writes	Moves	New state
Start_state	_	Y	Right	Halt
Start_state	0	_	Right	no_state
Start_state	1	_	Right	no_state
no_state	_	N	Right	Halt
no_state	0	_	Right	no_state
no_state	1	_	Right	no_state

Check if the tape is empty

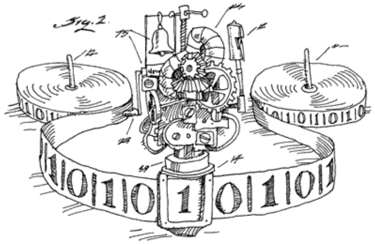


Current state	Reads	Writes	Moves	New state
Start_state	_	Y	Right	Halt
Start_state	0	_	Right	no_state
Start_state	1	_	Right	no_state
no_state	_	N	Right	Halt
no_state	0	_	Right	no_state
no_state	1	_	Right	no_state

- Simulator program:

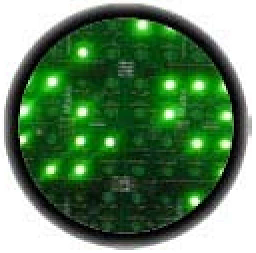
```
0             *  *  *  start_state      ; rename start state

start_state   _  Y  r  halt             ; if empty, write Y and stop
start_state   0  _  r  no_state         ; if tape has 0 or 1,  start erasing
start_state   1  _  r  no_state         ; while erasing, remember “no”

no_state      _  N  r  halt             ; tape is empty, write N and stop
no_state      0  _  r  no_state         ; keep erasing remembering “no”
no_state      1  _  r  no_state         ;keep erasing remembering “no”
```

Turing machine



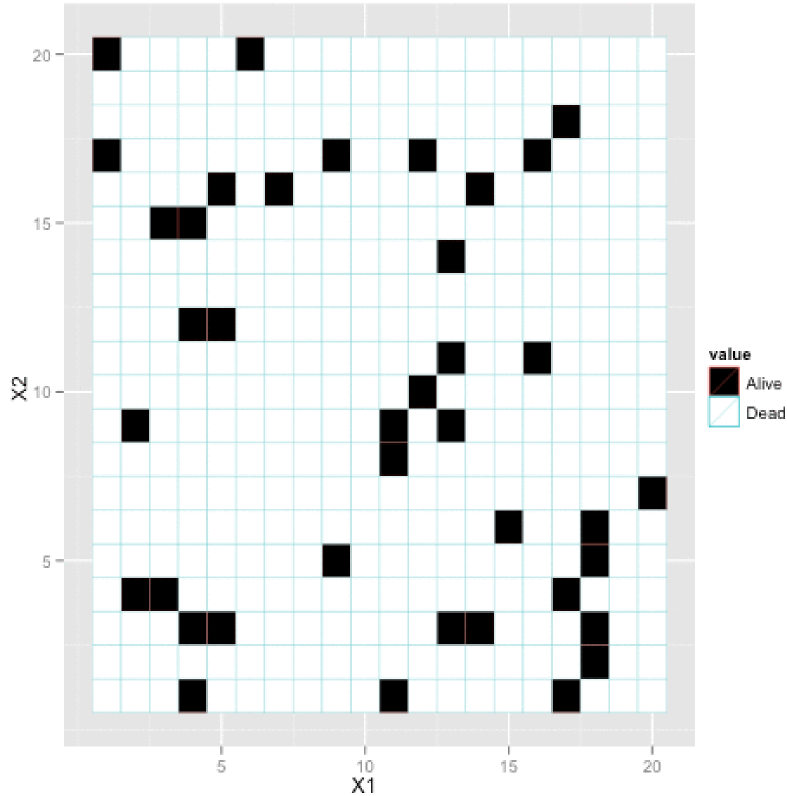
- Can do arithmetic (in binary)
  – see example of add 1
- Can do logic
  – Topic of the next class
- Can simulate any model of computation so far
  – Church-Turing thesis.
- Can have self-replicating programs
- Cannot solve some problems
  – “Am I lying”? “Is this true?”
  – “Will this computation ever stop?”

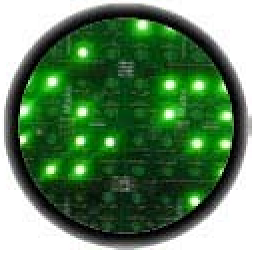
Does it mean nobody can solve them?



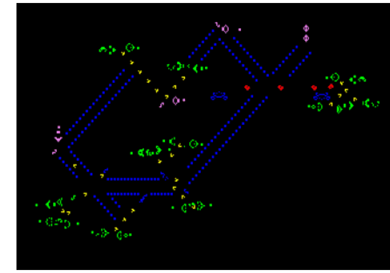
Conway’s game of life

- Rules of the Game of Life:
- Start with a board with a square grid
- Mark some grid cells as “live”
- At every step of the game:
  – Every live cell with less than 2 neighbours dies
  – Every live cell with more than 3 neighbours dies
  – A cell with exactly 3 neighbours becomes alive (is “born”).



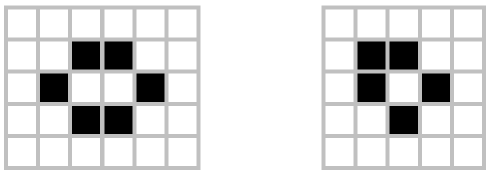


Conway’s game of life: what can it do?

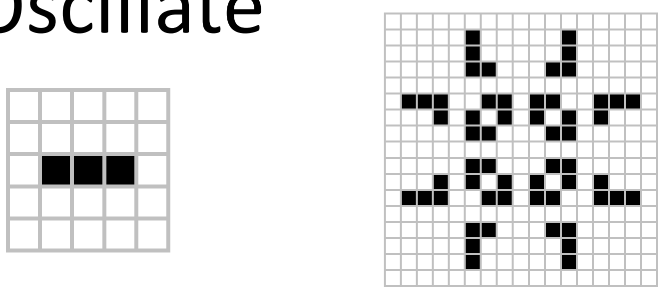


- Rules of the Game of Life:
- At every step of the game:
  - Every live cell with less than 2 neighbours dies
  - Every live cell with more than 3 neighbours dies
  - A cell with exactly 3 neighbours becomes alive (is “born”).

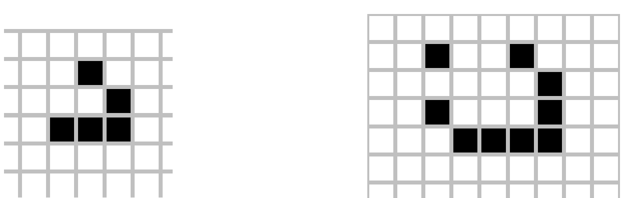
- Converge to a still pattern



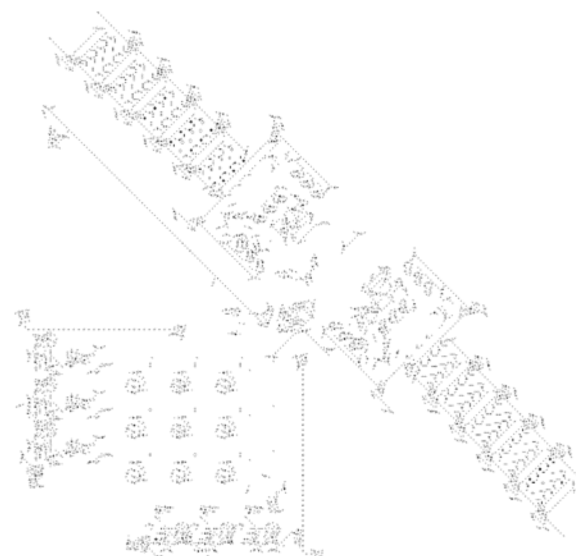
- Oscillate



- Create a moving pattern

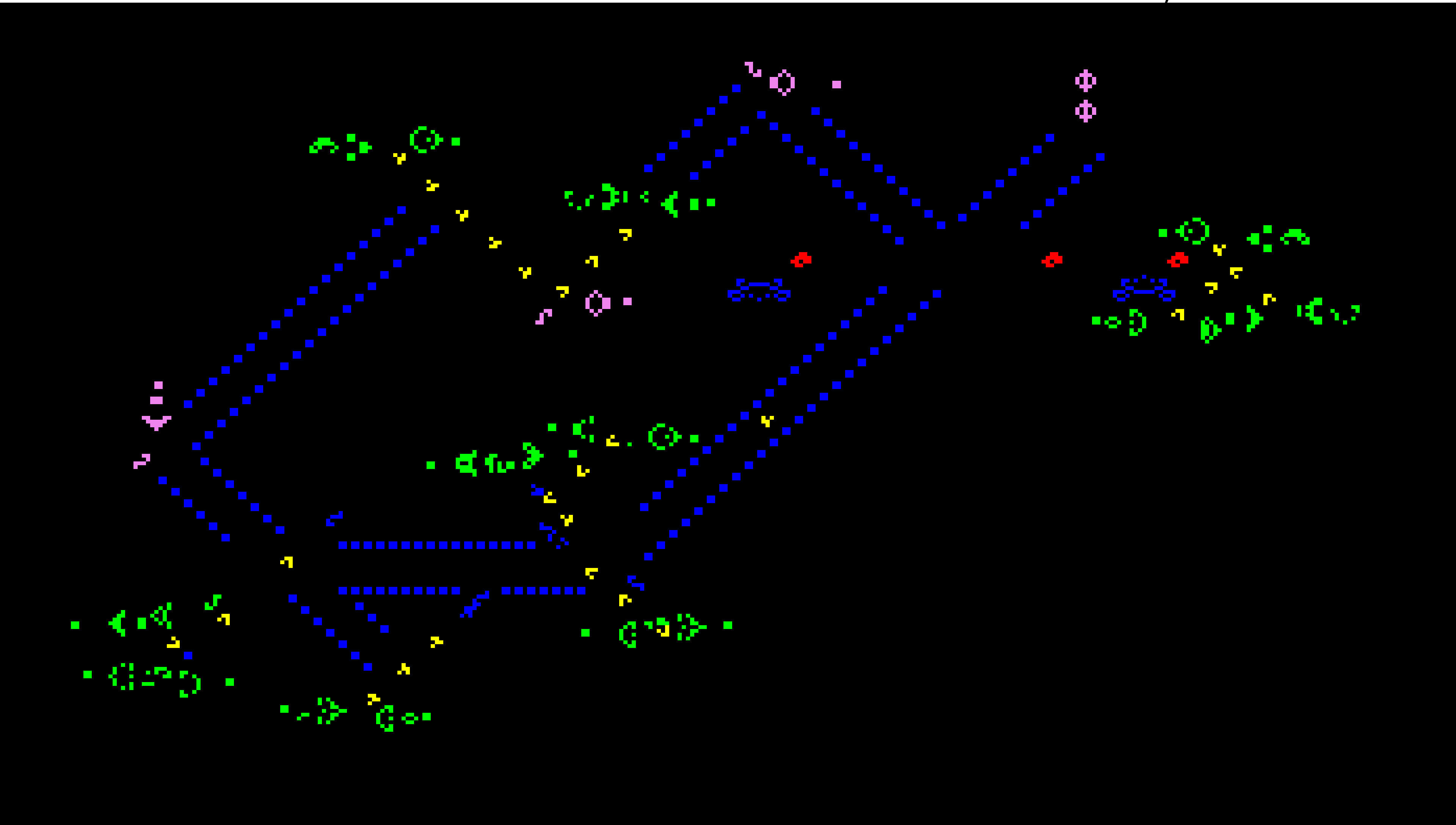


- Simulate a Turing machine



Conway's game of life: what can it do?

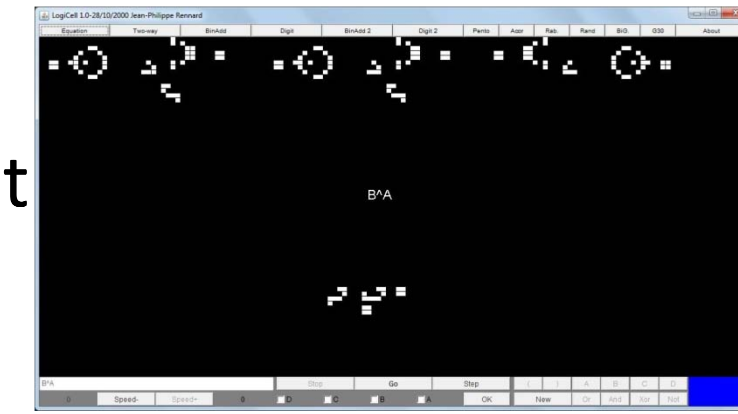
- At every step of the game:
  - Every live cell with less than 2 neighbours dies
  - Every live cell with more than 3 neighbours dies
  - A cell with exactly 3 neighbours becomes alive (is "born").

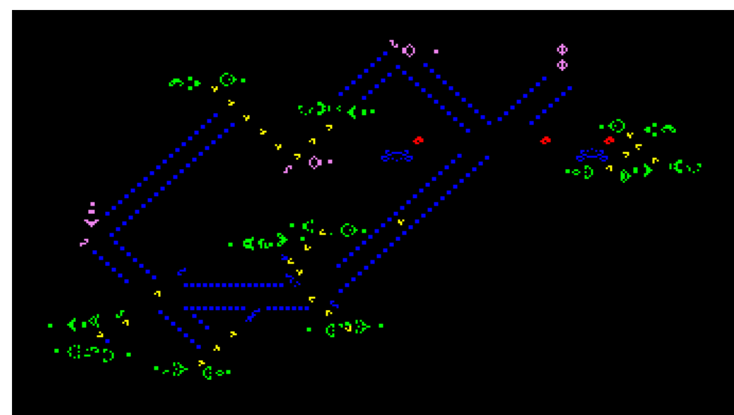


Conway's game of life: what does it mean to compute?

- Rules of the Game of Life:
- At every step of the game:
  - Every live cell with less than 2 neighbours dies
  - Every live cell with more than 3 neighbours dies
  - A cell with exactly 3 neighbours becomes alive (is "born").

- Start with a few cells lit up
- See if cells somewhere else light up
- Make it so they only light up if some condition holds
- Just like a Turing machine writing "Y" on the tape if some condition holds about its input

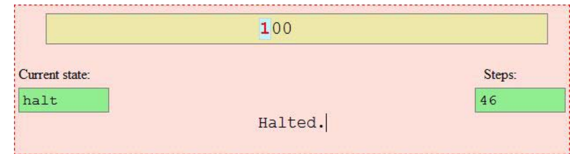




Simulators

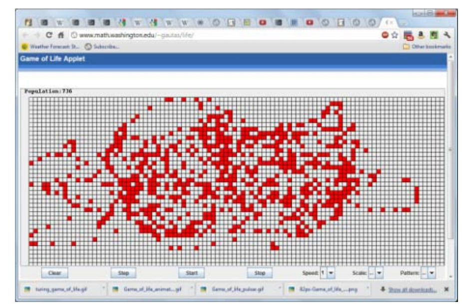
- Turing machine
  - Use this as the first line to give name to the start state

    0 * * * start_state

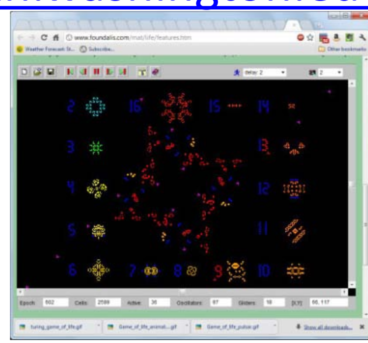


http://morphett.info/turing/turing.html

- Game of Life:
  - http://www.math.washington.edu/~gautas/life/



  - (with colours)



http://www.foundalis.com/mat/life/features.htm

  - LogiCell http://www.rennard.org/alife/english/logicellgb.html