ROKT

Coding question Game of Life with Wormhole

Background

The Game of Life, also known as Conway's Game of Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970. It is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input from humans. The "game" is a simulation of how life evolves based on a set of simple rules.

The Rules

The universe of the Game of Life is an infinite, two-dimensional orthogonal grid of square cells, each of which is in one of two states: alive or dead (populated or unpopulated). Every cell interacts with its eight neighbors, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

1. **Any live cell with fewer than two live neighbors dies**, as if by underpopulation.
2. **Any live cell with two or three live neighbors lives on** to the next generation.
3. **Any live cell with more than three live neighbors dies**, as if by overpopulation.
4. **Any dead cell with exactly three live neighbors becomes alive cell**, as if by reproduction.

These rules, which compare the behavior of the automaton to real life, can lead to a wide variety of patterns throughout the course of the game.

Game of Life with Wormhole

This is a special version of Conway game of life, where the 2D space contains wormholes that transport between different locations of the grid.

Wormhole Dynamics

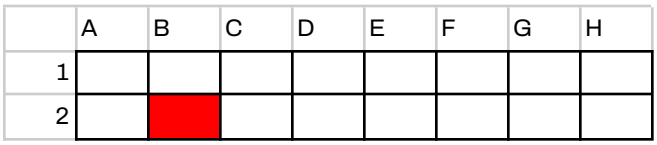
The underlying mechanism of these wormholes lies in the manipulation of the 2D space. Unlike the classic Game of Life, where cells interact solely with their immediate neighbors, this version introduces the concept of non-local interaction. Cells that are seemingly distant from each other can now become neighbors by virtue of being connected through a wormhole.

Horizontal Tunnel Bitmap

The structure and connectivity of these wormholes are encoded within a horizontal tunnel bit map. In this bitmap, pixels sharing the same color represent the boundaries of a wormhole. These boundaries act as portals, allowing cells to traverse from one location on the grid to another. A crucial rule of this bitmap is that for any given non-black RGB color value, there exist exactly two pixels with that color. This ensures that each wormhole has precisely two entry/exit points.

Example:

With a horizontal tunnel given by the following image



3								
4								
5								
6								
7								
8								
9								
10								
11								
12								

The modified topologic would be:

- A2's right neighbor is not B2 but G6 instead.
- B2's left neighbor is not A2 but F6
- G6's left neighbor is not F6, but A2
- F6's right neighbor is not G6 but B2
- A2 bottom right neighbour is not B3 but G7 instead.
- B2 bottom left neighbour is not A3 but F7

Vertical Tunnel Bitmap

Similarly, there is a vertical tunnel provided, which wraps around the vertical axis. The vertical orientation of the tunnel allows for movement or flow along the vertical axis, while the wrapping design ensures that this movement is contained within the tunnel structure.

Example:

With a vertical tunnel given by the following image

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								

The modified topologic would be:

- C2's bottom neighbor is not C3 but G7 instead.
- C3's top neighbor is not C2 but G6
- G6's bottom neighbor is not G7, but C3
- G7's top neighbor is not G6 but C2
- C2 bottom right neighbor is H7 and not D3
- C3 top right neighbor is H6 and not D2
- etc

Resolving map conflict

A single cell can be surrounded by a wormhole in more than one direction, this makes the handling of the corners ambiguous if there are more than 1 wormhole involved. When this occurs, the precedent order is top wormhole > right wormhole > bottom wormhole > left wormhole

Specific Instruction

Your task is to create an implementation of Conway's Game of Life with the added element of Wormholes, using your preferred programming language. We welcome the use of LLM or other AI power tools, including ChatGPT, Cursor, CoPilot etc.

Unzip the accompanying archive, which contains the problems to be solved. It consists of multiple directories, each holding the input data your program should operate on.

wormhole-interview-archive.zip

example-*/ contain the question and the answer.

Input

Each directory contains the following files:

- **starting_position.png:** A binary image specifying the initial state of the Game of Life board. White pixels represent live cells, and black pixels represent dead cells. All cells outside the boundary of the starting image are permanently dead and should remain so throughout the simulation.
- **horizontal_tunnel.png:** A color image where non black pixels indicate the positions of horizontal wormholes as described above.
- **vertical_tunnel.png:** A color image where non black pixels indicate the positions of vertical wormholes as described above.

All of these image files are guaranteed to have the same width and height.

Output

In the same directory as the input files, your program should generate three PNG files with the same dimensions as the starting_position.png:

- **1.png:** The state of the board after 1 iteration of the Game of Life with Wormholes.
- **10.png:** The state of the board after 10 iterations.
- **100.png:** The state of the board after 100 iterations.
- **1000.png:** The state of the board after 1000 iterations.

Submission

To finalize your submission, please follow these steps:

1. **Complete your program:** Ensure that your code within the 'srcs/' directory is fully functional and addresses all the requirements of the task.
2. **Compress your work:** Create a zip file containing the entirety of the input above, output together with the srcs/ directory
3. Submit to the Google Form provided in the email.

Additional Considerations:

- **Efficiency:** Strive to make your implementation as efficient as possible, as some input boards may be quite large.
- **Clarity:** Ensure your code is well-structured, documented, and easy to understand.
- It is very useful to have animate all the frames for debugging purpose

Evaluation

Your solution will be evaluated based on its correctness, efficiency, code quality, and thought process.