Marilyn Nguyen and Zelda Miller

Professor Paul Comitz

IS 247 Computer Programming

10 December 2023

Project Write Up

"The Game of Life" was developed by John Horton Conway in 1970. This game is a mathematical model and simulation that demonstrates a cellular automaton. Our project utilizes Java's Model View Controller framework to conduct Conway's Game of Life regulations and operations. For some background, the Game of Life is a simulation played on a 2D grid that represents the life of an organism and its interactions with other organisms as it evolves. This game begins by producing an initial configuration of a group of cells and observes how the group evolves in formation and number as it traverses the grid horizontally, vertically, or diagonally. Four rules control the functionality of this simulation; the first rule represents underpopulation, stating that any living cell with fewer than two live neighbors around it dies; the second rule represents overpopulation and says that any cell with more than three live neighbors dies; the third rule expresses that any cell with two or three live neighbors is unchanged and lives on; and the final rule states that any dead cell can come back to life it has three live neighboring cells around it.

In the implementation of Conway's Game of Life using the MVC framework, the model class encapsulates the game's logic, managing the grid state and applying rules for cell evolution. It handles data manipulation and notifies the controller of state changes. The view class represents the graphical user interface, displaying the grid and visualizing cell states. It receives updates from the model and relays user inputs to the controller. The controller class acts as an

intermediary, receiving user input from the view and updating the model accordingly. It ensures that changes in the model are reflected in the view, facilitating communication between the user interface and the underlying logic of the Game of Life. This MVC architecture ensures a clear separation of concerns, promoting code modularity and maintainability throughout game development.

The team strategically broke down the problem into smaller steps to address the project's challenges. Initially, the focus was on creating the view component to visualize the game board and gather user input for its dimensions. Conditional statements were employed to inspect cell placements near the board boundaries. A two-dimensional array stored the coordinates of randomly placed cells, and the model class governed the Game of Life rules through if and else statements within the MVC framework.

Delimiter characters, such as "=", were incorporated between generations for visual separation and border creation. The view class generated the 2D game board and cell placements, while the controller component received user input, continuously updating the board's view as cells evolved. One team member focused on programming the Controller element, while the other worked on the View portion. Collaboration between both members was integral to coding the Model component effectively. An iterative development process, strategic problem-solving, and collaboration led to the successful implementation of Conway's Game of Life using the MVC framework.

As the entry point for our Game of Life program, the "Main" class facilitates user interaction and initiates game setup. It utilizes a Scanner to prompt the user for input, collecting information such as the number of rows and columns for the game board, the desired number of generations, the delay time between generations, and the chosen cell pattern file. These user

inputs are then used to instantiate a "Controller" object, which directs the interactions between the "Model" and "View" classes. The "Main" class invokes the "runGame" method in the "Controller" class, initiating the Game of Life simulation based on the user-provided parameters. Overall, the "Main" class is pivotal in configuring and launching Conway's Game of Life simulation, offering a user-friendly interface for setting up and visualizing the evolving cellular automaton.

The "Model" class encapsulates the core logic and data structures for the Game of Life simulation. Its constructor initializes a two-dimensional boolean array representing the grid of cells, and the "initializePattern" method reads a specified pattern file to set the initial state of the grid. The "updateGrid" method applies Conway's Game of Life rules to transition the current generation to the next, determining the fate of each cell based on its neighbors. Additionally, the class includes a "getGrid" method to retrieve the current state of the grid and a "countNeighbors" method to calculate the number of live neighbors for a given cell. The "Model" class acts as the computational engine, facilitating the evolution of the cellular automaton and providing a foundational structure for the overall Game of Life simulation.

The "View" class is responsible for presenting the Game of Life simulation to the user through the console interface. Its "printGrid" method displays the current state of the grid, including borders, using a simple ASCII representation with asterisks denoting live cells. The "displayDelimiter" method outputs a visual delimiter to separate successive generations, enhancing the readability of the simulation. The "clearScreen" method provides a means to clear the console screen, creating a clean slate for each new generation display. Focusing on visual representation and the formatting of the simulation, the "View" class ensures a clear and organized presentation of the evolving cellular automaton is shown to the user.

Orchestrating the "Model" and "View" components, the "Controller" class manages the flow of the Game of Life simulation. Its constructor initializes instances of the "Model" and "View" classes, establishing a connection between the computational and visual aspects of the simulation. The "runGame" method coordinates the execution of the simulation over multiple generations, periodically updating the grid state, printing it using the "View" class, and introducing a time delay between generations for user-friendly visualization. The "Controller" class encapsulates the overall control logic, ensuring seamless interaction between the model's computational rules and the view's presentation, thereby facilitating a cohesive and engaging experience for the user interacting with the Game of Life simulation.

In conclusion, our implementation of Conway's Game of Life is a testament to the synergy between mathematical abstraction and computational modeling. We've successfully encapsulated the elaborate cellular journey using Java's Model-View-Controller framework. The Model class governs the logic and rules, orchestrating the dynamic interplay of cells on a 2D grid. The View class inserts these machinations into a visually digestible form, offering clarity and organization through ASCII art. Orchestrating this harmonious interaction is the Controller class, ensuring a seamless flow between the underlying computational engine and the user interface. Collaborative problem-solving, strategic breakdown of challenges, and an iterative development process have shaped a program that captures the elegance of Conway's creation and exemplifies the modularity and maintainability inherent in the MVC architecture.

Works Cited

Caballero, L., Hodge, B., & Hernandez, S. (2016, May 17). *Conway's "Game of life" and the epigenetic principle*. Frontiers. https://www.frontiersin.org/articles/10.3389/fcimb.2016.00057/full

Johnston, N. (2019, October 9). Conway's Game of Life. https://conwaylife.com/