```
#include <iostream>
#include <vector>
#include <cstdlib>
#include <unistd.h>
#include <fstream>

using namespace std;

class Universe {
private:
    int rows, cols;
    vector<vector<bool>> grid;
    vector<vector<bool>> next_grid;

public:
    Universe(int r, int c) : rows(r), cols(c), grid(r, vector<bool>(c)), next_grid(r, vector<bool>(c)) {}

    void initialize(double live_percentage = 0.5) {
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                grid[i][j] = (rand() % 100) < (live_percentage * 100);
            }
        }
    }

    void loadPatternFromFile(const string& filename) {
        ifstream file(filename);
        if (file.is_open()) {
            file >> rows >> cols;
            grid.resize(rows, vector<bool>(cols));
            next_grid.resize(rows, vector<bool>(cols));
            string line;
            for (int i = 0; i < rows; ++i) {
                file >> line;
                for (int j = 0; j < cols; ++j) {
                    grid[i][j] = (line[j] == '1');
                }
            }
            file.close();
        } else {
            cerr << "Unable to open file" << endl;
        }
    }
```

```
void reset() {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            grid[i][j] = false;
        }
    }
}

int count_neighbors(int x, int y) {
    int count = 0;
    for (int i = -1; i <= 1; ++i) {
        for (int j = -1; j <= 1; ++j) {
            if (i == 0 && j == 0) continue;
            int nx = x + i, ny = y + j;
            if (nx >= 0 && nx < rows && ny >= 0 && ny < cols) {
                count += grid[nx][ny];
            }
        }
    }
    return count;
}

void next_generation() {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            int neighbors = count_neighbors(i, j);
            if (grid[i][j]) {
                next_grid[i][j] = (neighbors == 2 || neighbors == 3);
            } else {
                next_grid[i][j] = (neighbors == 3);
            }
        }
    }
    grid.swap(next_grid);
}

void display() {
    cout << "\033[2J\033[1;1H";
    cout << '\n';
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            cout << (grid[i][j] ? 'O' : '.');
```

```
            }
            cout << '\n';
        }
        cout << '\n' << string(cols, '-') << '\n' << '\n';
    }

    void run(int generations) {
        for (int i = 0; i < generations; ++i) {
            display();
            next_generation();
            usleep(200000);
        }
    }
};

int main() {
    Universe universe(20, 50);
    int choice;

    while (true) {
        cout << "Menu:\n";
        cout << "1. Initialize with random pattern\n";
        cout << "2. Load pattern from file\n";
        cout << "3. Run simulation\n";
        cout << "4. Reset\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        if (choice == 1) {
            double live_percentage;
            while (true) {
                cout << "Enter the percentage of live cells (0-100): ";
                cin >> live_percentage;
                if (live_percentage >= 0 && live_percentage <= 100) {
                    break;
                } else {
                    cout << "Invalid input. Please enter a number between 0 and 100." << endl;
                }
            }
            universe.initialize(live_percentage / 100.0);
        } else if (choice == 2) {
            string filename;
```

```
            cout << "Enter the filename: ";
            cin >> filename;
            universe.loadPatternFromFile(filename);
        } else if (choice == 3) {
            int generations;
            while (true) {
                cout << "Enter the number of generations: ";
                cin >> generations;
                if (generations > 0) break;
                cout << "Invalid input. Please enter a positive number." << endl;
            }
            universe.run(generations);
        } else if (choice == 4) {
            universe.reset();
            cout << "Universe has been reset." << endl;
        } else if (choice == 5) {
            cout << "Exiting..." << endl;
            break;
        } else {
            cout << "Invalid choice. Please try again." << endl;
        }
    }

    return 0;
}
```