**CS273 Final Project - Speciification Document**

**Team members: Anujin Munkhbat, Amon R Sthapit, Abdul Haq**

**Project name: Game of Life**

* **Requirement specification:**

Game of life is a zero-player game, meaning that its evolution is determined by its initial state. We’ll create initial configuration with some rates determined by user input and (simulate) observe how it evolves. Basic rules are:

1. Any live cell with fewer than two live neighbors dies, as if caused by under population.
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 a live cell, as if by reproduction.

Our purpose is to modify the rule of reproduction by using hashmap and queues. Inheriting from our base class (NewGame), we’ll create 2 child classes, which will modify certain rules using polymorphism.

* **Use Cases:**

**main():**

1. **Prompt the user for the type of game to simulate** 
   1. **Use case 1 - By\_Generation:**

Prompt the user for zombie\_mode\_wait\_rate

// number of generations to keep a dead cell in a zombie mode until it’s reproduced

Initialize the matrix //initially all cells = 0 (dead)

Call **set\_state()**  // initial positions of live cells

Call **iterate()** // calls every rule

//modified rules:

**rule\_1():**

Initialize **live = 0** //count of live neighbours

For loop for rows

For loop for columns

**if** current cell is alive: // cell == 1

live = live\_count() // get number of live neighboours

If  **live < 2:**

Set cell to **0** //dead

Add to the hashmap of dead cells

//only those who died during the simulation

// {generation : [row, col]}

**rule\_3():**

Same as rule\_1() except check for **live > 3**

**rule\_4():**

**Get\_generation = curr\_gen - zombie\_mode\_wait\_rate**

**Set all cells in that generation to 1**

* 1. **Use** **case 2** **-** **By\_Queue:**

Prompt the user for resurrection\_rate //num of cells to reproduce in each generation

Initialize the matrix

Call **set\_state()**

Call **iterate()**

//modified rules:

**rule\_1() and rule\_3():**

Same as above except we’ll add cells that died during the current generation

to a queue of zombie mode.

**rule\_4():**

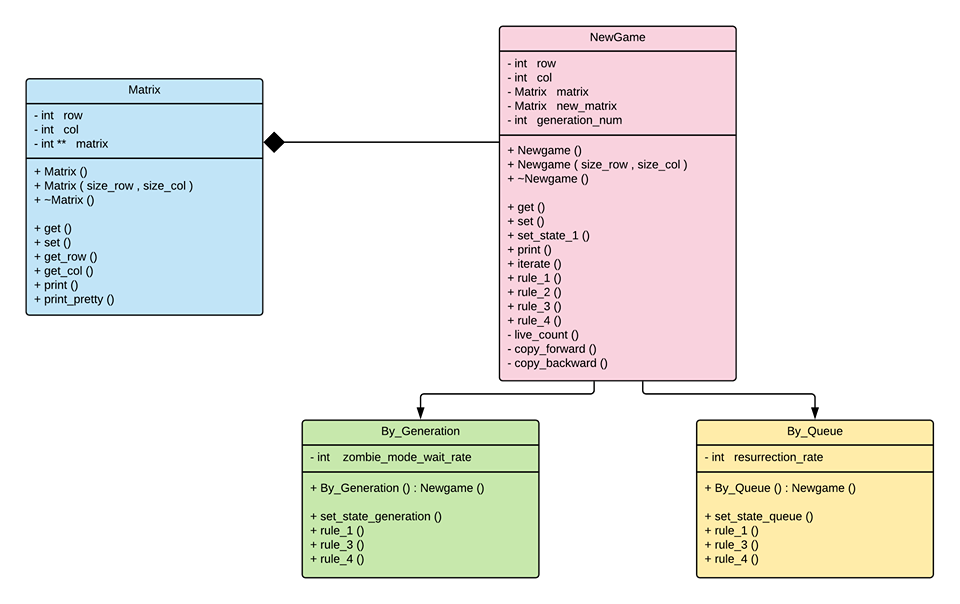
N = resurrection\_rate

Reproduce n(or less) cells from the queue

1. **Prompt the user for number of generations to simulate**
2. **Create a game object**
3. **game-> set\_state()**
4. **For loop:**

game-> print()

game->iterate()

* **UML Diagram:**

**Print()** in NewGame calls **print\_pretty()** from Matrix class. This allows us to show the simulation in an animated way.

**set\_state()** sets the initial configuration of the live cells

**Iterarate()** calls all 4 rules to apply the rules simultaneously

**live\_count()**  returns a number of alive neighbours of the cell

**Copy\_forward()** copies the state of the current matrix to the next one, which will be the next generation

**copy\_backward()** sets the next\_matrix to the current one