Game of Life

Team Details

1.	Rupesh	160050042
2.	Ishan Tarunesh	160050054
3.	Sushant Tarun	160050055

Problem Statement

Game of Life is an example of **cellular automaton** created originally by John Conway. The simulation has a 'world' (**infinite grid of cells**) and an **evolutionary-law**. Given any initial state of live and dead cells, the law can be used to decide the **future state** of the cells. We have attempted to adapt this simulation in racket and also add more features (like **mutable-law**, **hexagonal** cells instead of **square** cells).

Overall Design (Structure of Program)

1. **main.rkt -** This file contains all **main** and **helper functions** for evolution of one state to other. We have implemented the input as **Hash Table** with **row number** as **hash keys**. This ensures easy and fast lookup by the **updater** function. The **neighbour** function also implements a **Hash Table** with **number of occurrence** of each coordinate as the **Hash key**. The **loop runs over only the live cells** instead of the whole board in each cycle hence **efficiently and quickly** calculates the next generation. Each cell on the matrix can be represented by a struct point which has three fields, **i (row no) j (col no)** and **state (dead or alive).**

2. **run.rkt -** Contains all graphics(**frame, canvas, buttons, panel, choice, panel** etc) of the program. Canvases are **overridden with on-click events** and **images** are used as **buttons.**

3. **Sample Inputs -** Comprising of two files named “test_cases_square.rkt” and “test_cases_hex.rkt”. Inputs for the square cell are implemented using hash table while that of hexagonal board is made using **2D-vectors**. It includes some large test cases like **puffer-train, vacuum-cleaner** etc which were **parsed** from their .rle file using a **C++ program** we made.

Major Features

1. **User Input** : - The user can **click on the simulation canvas** to create his own **initial pattern** and further simulate it to observe its evolution. By enabling click event on canvas, we were able to implement click input on square-board (small and large) and hex-board.
2. **Frame Rate** - User can click on buttons “**Fast**” and “**Slow**” in the simulation frame to increase and decrease the rate of simulation respectively. This was implemented using a global variable - **time** which stores the **delay** between execution of simulator.
3. **Mutable Evolutionary Law** - User will be able to modify the rules of evolution **(initially B2/S2,3)** in the settings page.
4. **Cell Shape** - In settings page, **cell shape** can be changed to **hexagon**.
5. **Statistical Features** - Used **Object/Class** to implement **TotalAlive** and **MaxPopulation** functions. MaxPopulation function keeps a record of peak population the simulation reached till then.
6. **Test Case Parsing** - Patterns from an existing simulator golly.exe were parsed using a C++ program to be used as large inputs in our program.
7. **Music -** Music plays in the background when simulation is started, **when board becomes empty the music stops**.

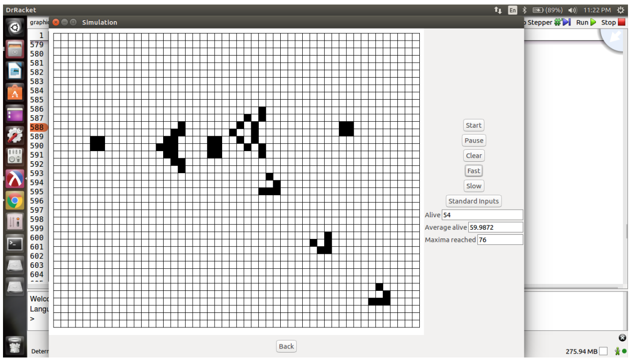
Limitations and Bugs

1. Program freezes when fast button is pressed multiple times.
2. Hexagonal Board couldn’t be implemented as a Hash Table because on-click event was getting delayed.

Sample Input/Output

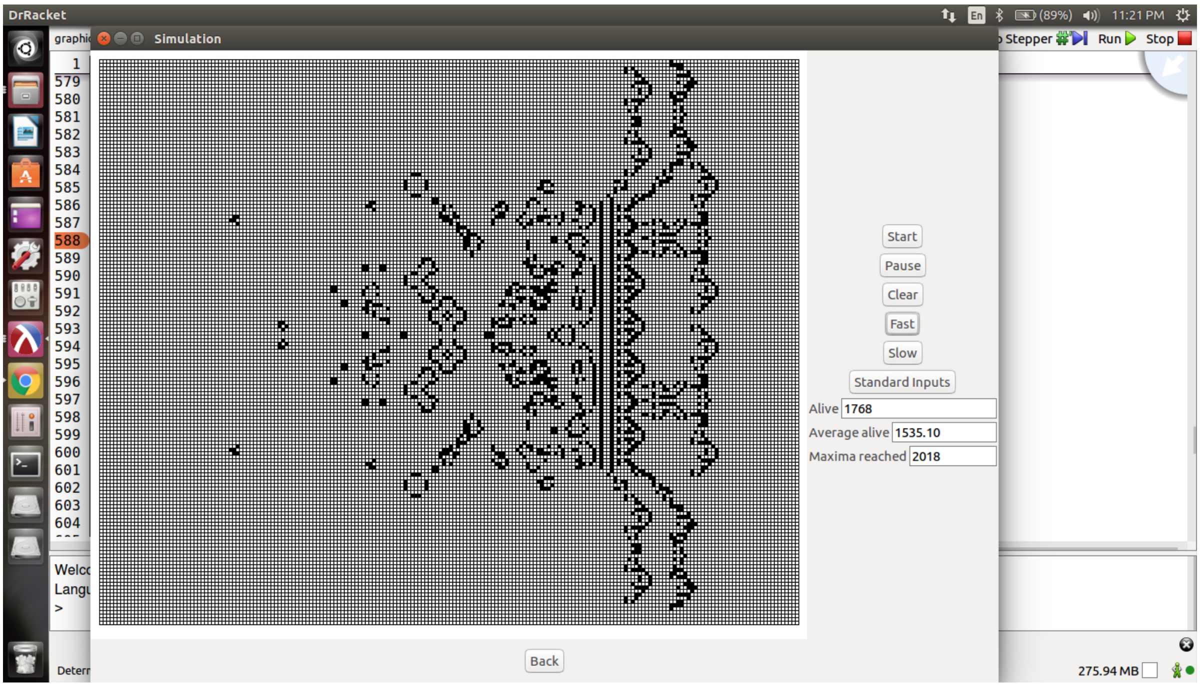
3. Gosper-Glider Gun

   This test case involves two **guns** moving towards each other and bouncing-off **walls**. After in cycle a glider is sent off diagonally as shown in figure.



Alive 54
Average alive 59.9872
Maxima reached 76

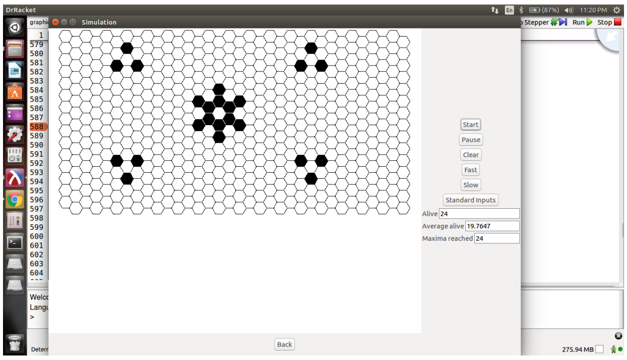
4. Puffer Train



Alive 1768
Average alive 1535.10
Maxima reached 2018

   This is a stable **train**-like test case which moves forward at a constant speed leaving behind rakes.

5. Hexagon-Example - These include collection of many oscillatory test cases like **star, tri-star** etc on a hexagonal board with rule **(B2/S3,4)**.



Alive 24
Average alive 19.7647
Maxima reached 24