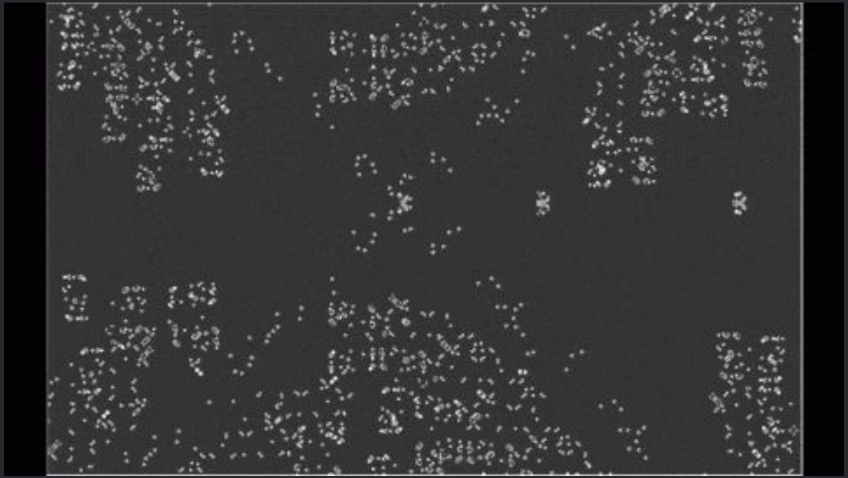
CONWAY'S GAME OF LIFE IN 3D SPACE

Gourav Pullela

WHAT IS CONWAY'S GAME OF LIFE

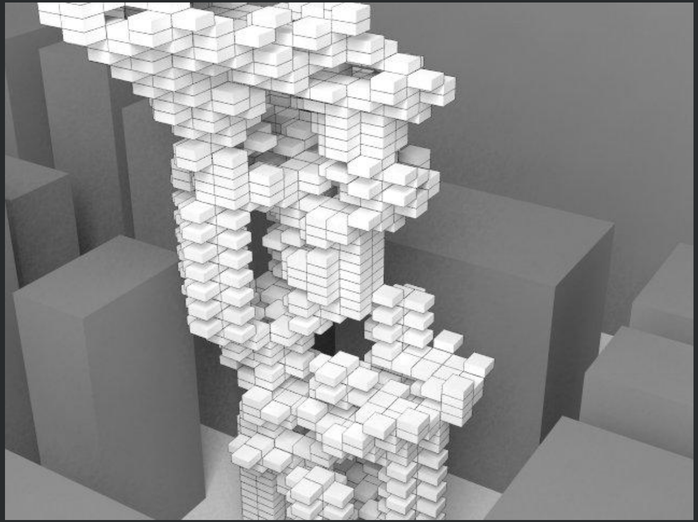
Conway's Game of Life is a cellular automaton devised by John Conway.

The game of life is a zero-player game and is determined by its initial state.

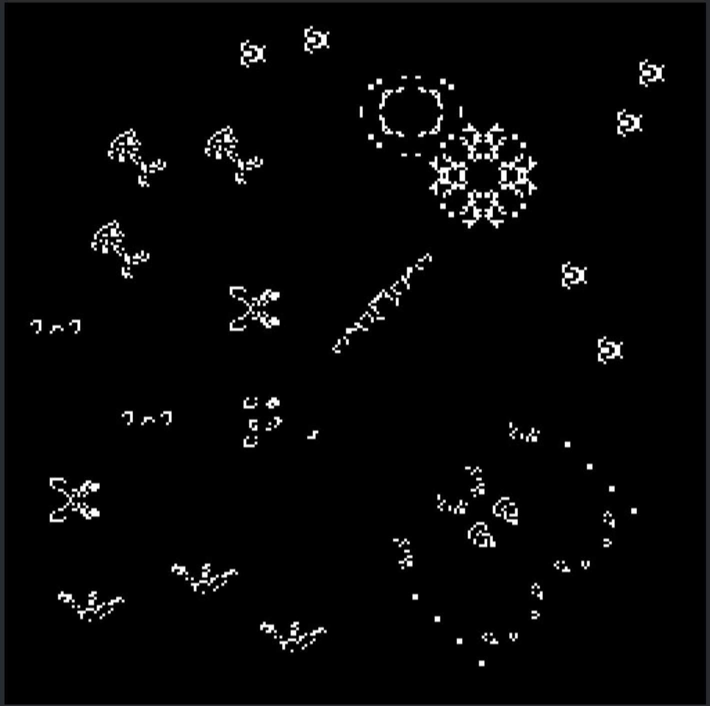


PROJECT GOALS

The goal of this project was to implement Conway's Game of Life as a simulation in three dimensional space.



THE RULES OF THE GAME

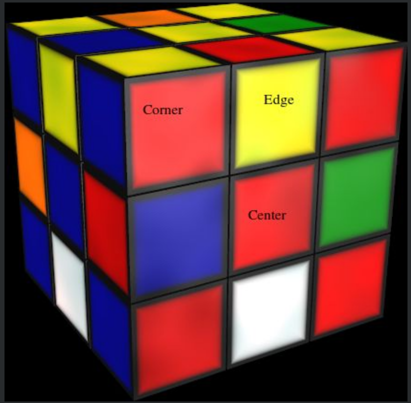


1. Any live cell with 2 or 3 live neighbours survives.
2. Any dead cell with 3 neighbors becomes a live cell.
3. All other live cells die before the next cycle.

IMPLEMENTATION OF THE GAME

I chose to use OpenMP C to implement the game in 3D.

The reason I chose to OpenMP is because it is easy to use and allowed scalability for code when I converted the 2D game to 3D.



MODIFIED RULES

- Any live cell with less than 6 or greater than 11 live neighbours dies.
- Any dead cell with more than 7 and less than 12 live neighbours lives.
- Each cell has a total of 26 neighbors, which can be horizontally, vertically, or diagonally adjacent to that cell.

OUTLINING THE TEST CASE

- Used ***srand(time(NULL))*** to seed rand() based on current time

- A 3D Array of ***20*20*20*** was used as the test set.

- Approximately ***33%*** of the matrix has live cells when initially populated using the rand() function.

- Simulation cycles are run until the total population is ***0***.

CODE SNIPPET

```
//Returns number of living neighbors
int numAlive(struct Cell game[N][N][N], struct Cell point, struct Vector* n) {

   int i,alive = 0;
   #pragma omp parallel for
      for(i=0; i < 26; i++) {
         if((-1 < n[i].x && n[i].x < N) && (-1 < n[i].y && n[i].y < N) && (-1 < n[i].z && n[i].z < N)) {
            if(game[n[i].x][n[i].y][n[i].z].doa == 1) {
               alive += 1;
            }
         }
      }

   return alive;
}
```

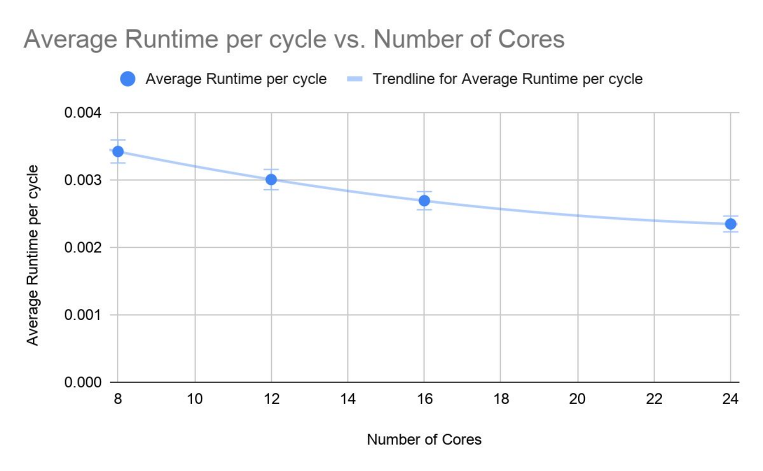
This function is used to find the number of living neighbours around a cell, which can in turn be used to determine its state.

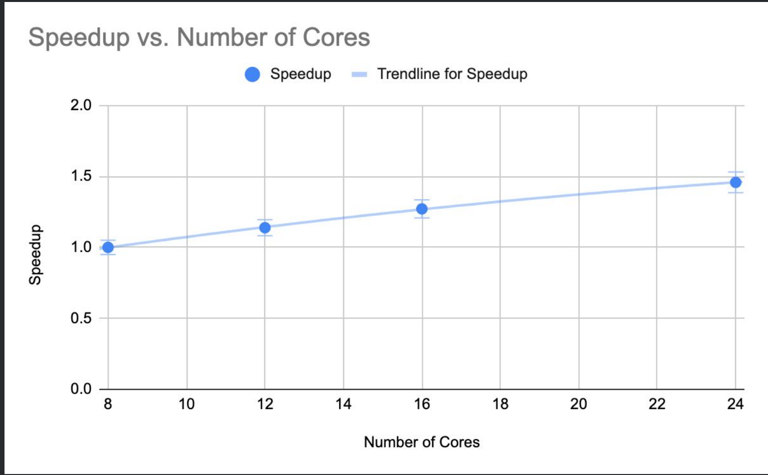
CODE SNIPPET

```
//Simulation
while(count > 0) {
   #pragma omp parallel for collapse(3)
      for(i=0; i < N; i++) {
         for(j=0; j < N; j++) {
            for(k=0; k < N; k++) {
               input = neighbors(game[i][j][k]);
               alive = numAlive(game, game[i][j][k], input);
               if(game[i][j][k].doa == 1 && (alive < 6 || alive > 11)) {
                  game[i][j][k].doa = 0;
                  count -= 1;
               }
               if(game[i][j][k].doa == 0 && (alive > 7 && alive < 12)) {
                  game[i][j][k].doa = 1;
                  count += 1;
               }
               if(count > maxcount) {
                  maxcount = count;
               }
            }
         }
      }
   steps += 1;
}
```

This code block is the simulation segment of the program.

PERFORMANCE WITH WEAK SCALING





8000 Cell Simulation with ~ 33% initial population.

FURTHER IMPLEMENTATION/ADDITIONS

- Studying population density in real time.
- Visualization of the simulation.
- Studying the average lifespan of cells and how modification of the rules affects it.
- Studying growth and decay rate of the cell environment.
- Implementation of the project in CUDA C.

THANK YOU

Do you have any questions?

gpullela@iu.edu