**Conway’s Game of Life**

By Christopher-Robin Ebbinghaus

**Change History**

|  |  |
| --- | --- |
| Version | Description |
| 0.1.0 | First Draft Filling out All Information |

1. [Technical Specifications](#TechnicalSpecifications):
   1. [Game Engine](#GameEngine)
   2. [Programming Language](#ProgrammingLanguage)
   3. [Integrated Development Environment](#IDE)
   4. [Source Control Software](#SourceControl)
   5. [Project Management Tools](#ProjectManagement)
   6. [Third Party Libraries](#ThirdParty)
   7. [Additional Software and Resources](#AdditionalSoftware)
2. [Timeline](#Timeline)
   1. [Milestones](#Milestones)
3. [Overview](#Overview)
   1. [Gameplay](#GamePlay)
   2. [Logic](#Logic)
      1. [Game Rule Specifications](#GameRules)
4. [Game Architecture](#GameArchitecture)
   1. [Core Data Structures](#DataStructures)
   2. [Class Overview](#ClassOverview)
5. [User Interface](#UI)
   1. [Game UI Elements](#UIElements)
6. **Technical Specifications:**
   1. Game Engine:

The Game Engine of choice is the AIE Bootstrap Engine because it provides a very low-level interface allowing for efficient and low overhead performance of the Game. The open source nature also allows me to make modifications and additions if needed to extend or improve the performance for certain aspects.

* 1. Programming Language:

C++ was Chosen as the Primary Language for its speed and its use in the Bootstrap Engine. Its Efficiency and Low-Level status also allow for Bigger Playing fields.

* 1. Integrated Development Environment   
       
     As Development Environment I have Chosen Visual Studio over Eclipse or co, because of its superior debugging experience and its IntelliSense. The Current version is 2017 but in future there Might be an Upgrade to 2019.
  2. Source Control Software

For source Control I have chosen Git Kraken because of its beautiful UI and exceptional functionality. Since its extending Git’s functionality on the lowest level any Git supported Software can be used Instead allowing for another Developer to use A Software more suited to them.

* 1. Project Management Tools  
       
     As for my Project Management Tool I have chosen Glo. Made by the same Company that created Git Kraken it’s a List creation Tool like Trello or Evernote that seamlessly Integrates into Git Kraken.
  2. Third Party Libraries  
       
     The Only Third party Library used will be the Bootstrap Engine and of its Dependencies.
  3. Additional Software and Resources

I will use Some form of Icon Pack to get Sprites for Buttons but nothing it finalized as of Version 0.1.0

1. **Timeline:**
   1. Milestones
2. Base Simulation working in a 3x3 Grid
3. Addition of Drawing Pixels to Grid
4. Clear, Shuffle and Pause/Play Buttons and Functions Added
5. Resizing of the Field (scrolling in and out)
6. Automatic Sizing of the Playing field to span all monitors
7. Positioning to allow for Multi Monitor Fullscreen Drawing
8. Efficiency Increases to allow for a Field to Pixel Ratio of 1:1
9. **Overview:**
   1. Gameplay  
        
      Gameplay will feature a small Bar of UI buttons to Interact with the Field. Functionalities will include Clearing the Field, Randomizing the Cells, Manually Drawing cells and Playing/Pausing the simulation. The game will follow the Simulations Specifications and will run stably at 30fps.
   2. Logic  
        
      Most of the Game Logic will be in the Simulation Specifications due to the Nature of the Projects. If The simulation is currently playing all buttons except the play/pause one will be Dimmed to Indicate that no changes can be made. The Ui Buttons will behave as follows:

* **Randomizer:** The Random Button Will Loop over every Element in the Field and set its state depending on a 50/50 chance.
* **Clear:** The Clear button will Loop over Every Element and set its state to false to clear the entire Playing Field
* **Pencil:** The Pencil mode will be a Dual-Purpose button. When not Selected it will activate the Drawing Mode in which the user can Draw on the field to make changes to the field. When Already selected it will switch between Pencil and Brush Mode.
  + Pencil will draw every Pixel that the Point of the Mouse touches as if colouring a single pixel at a time. Useful for drawing exact shapes to create specific scenarios.
  + Brush Will fill every pixel in a certain radius to more easily allow for the user to fill larger areas.
* **Play/Pause:** The Play/Pause button will toggle if the Game is Playing or Paused. This will allow the Player to draw or make changes to the Field.
  + 1. Game Rule Specifications:  
         
       The Core game Rules are Based off the Original Simulation Specifications of the Conway’s game of life Experiment. Which can be found Here. The Basic Rules are:
* Any Active Cell that is surrounded by 2 or less Active Cells Dies.
* Any Active Cell that is surrounded by exactly 3 Stays Alive
* Any Active Cell that is Surrounded by more than 3 Active cells Dies.
* Any Dead Cell that is surrounded by exactly 3 Active cells Comes to Life

These Rules Create an Interesting Experiment Environment allowing for Study of “Human” behaviour in big groups. While its Scientific Significance is Debatable from a Game Aspect it Proposes an Interesting Experience allowing for an Infinite Sandbox experience whereby there are more than 2^xy Board Possibilities and even more Simulation Outcomes.

1. **Game Architecture:**
   1. Core Data Structures:

At its Core the Game Relies on Arrays.

This can either be achieved by using a Singular Large XY sized Array using complex Math to retrieve values at certain X, Y positions.

Or it can be Implemented using a Multi-Dimensional Array that has X, Y accessors. This would Increase Code readability and make for a more enjoyable Coding Experience at the Cost of Performance.

I will Likely need to do Some Testing and Revise this section as the Project Continues.

* 1. Class Overview:

Game Class:

GameField

Pen Tool

RenderingClass

Input Logic

GameField Class:

2 2D Arrays of Bools

Conways Rule Logic

Pointer for Drawing Array

Pen Class:

Coordinates for X and Y

Functions to Draw and Erase a certain Pixel

Rendering Class:

Function to Render the Field to Screen

1. **User Interface:**
   1. Game UI Elements:

