# Worksheet for Week 2 – Rule BASED SYSTEMS

## Overview

### Objectives:

1. Play and understand a RBS Example
2. Finish the ruleset for the Game of Life
3. Extend the ruleset

### Bonus Objectives

1. Build a BOIDS example

## Preliminary

Primarily this tutorial work will require the use of a C++ IDE. Visual Studio is the recommended software to use although any C++ development environment should be sufficient.

There is a Visual Studio Project on MyLearningSpace to use for this week’s work. A link to this will be available on MyLearningSpace, titled ‘Rule Bases Systems – C++’.

If you need help setting up the software, we have a tool called AppsAnywhere that allows you access to some of the software we use. You can use this to install some of the software we use. <https://myapps.abertay.ac.uk/>

## Getting Started

Go to the following website and experiment with the Game of Life. <https://bitstorm.org/gameoflife/>

The site gives you a window to play the game and an explanation of the rules. To summarise:

**For a space that is 'populated':**

Each cell with one or no neighbours dies, as if by solitude.

Each cell with four or more neighbours dies, as if by overpopulation.

Each cell with two or three neighbours survives.

**For a space that is 'empty' or 'unpopulated'**

Each cell with three neighbours becomes populated.

Use the link in MyLearningSpace to access the *SFML\_RuleBasedSystem.zip* assignment. Download the assignment onto your machine. Open the Visual Studio Project.

The project contains a folder named Executable that runs a standalone version of the program if you want to play with game without recompiling the entire project.

## Finish the Ruleset for SFML example

The rules in the *SFML\_RuleBasedSystem* project is similar to the Game Of Life but it starts with different rules.

**For a space that is 'populated':**

Each cell has a chance to move to a unoccupied adjacent plot (cannot move off map)

Each cell with more than one neighbour dies

**For a space that is 'empty' or 'unpopulated'**

Each cell with more than one neighbour becomes populated

Unlike the regular version of the Game of Life, there is no death by loneliness rule.

1. Open the project in Visual Studio and compile the program
2. Play the game and experiment with shapes
3. Edit the function on line 77:

Currently it is very crudely done, if a grid space is occupied and it has more than one neighbour, then it becomes unoccupied

If it is unoccupied and has more than one neighbour, then it becomes populated

Edit this code so it matches with the regular version of *Game of Life.*

1. Add the death by loneliness rule

Occupied nodes with no neighbours becomes unoccupied

## Extend the ruleset

Extend the ruleset to accommodate the following rules:

1. Random spawning once every five cycles:

Find an unoccupied node in the grid and populate it

1. Random death every six cycles

Every six cycles of the game, find a populate node and kill it

1. When the number of occupied nodes is over 75% the size of the grid, kill half of all occupied nodes

Add your own rules or edit the ones you currently have as you see fit.

## Bonus Objectives

This is very open ended but gives you an opportunity design and build an RBS system of your own. You need to create a visual representation of a BOIDS system.

Boids is an RBS that controls the movement of a swarm/crowd/group.

Classically the flock does the following:

* Alignment – Unit in the group will align itself with the average of the rest of the flock
* Cohesion – Unit will steer towards the average position of the rest of the flock (centre of mass)
* Separation – Units will maintain a distance between other flock members

### Design the system

Design what the database will contain and the rules the system would contain. Consider how each member of a flock would interact with another, what data they will require, what necessary data needs saved etc.

Write a section of pseudocode to explain how the program would run (you don’t have to worry about specifics of drawing or rendering). Explain how the data is written to memory, how the flock access the data, how the RBS system reads and executes the data and its rules.

### Write the program

If you feel very adventurous, code the Boids system into a real time simulation. You can use any tools that you want, but it may be easier to use a Game Engine (Unity leaps to mind).

There is an example (with some very clear issues) available on MyLearningSpace that you can use as guidance.

*Boids\_in\_Unity.zip*