Contents

[0 Introduction 2](#_Toc129637689)

[1 The System 3](#_Toc129637690)

[1.1 The Simulation 3](#_Toc129637691)

[1.2 Environment Settings 5](#_Toc129637692)

[1.3 Agent Settings 6](#_Toc129637693)

[1.5 Diagnostics 8](#_Toc129637694)

[1.5 Terrain Generation 9](#_Toc129637695)

[1.6 System Controls 10](#_Toc129637696)

[2 Known Issues 11](#_Toc129637697)

[2.1 Error: could not open ‘%FLATLAND\_LOCATION%\jre\lib\jvm.cfg’ 11](#_Toc129637698)

[2.1 Controls do not respond. 11](#_Toc129637699)

# 0 Introduction

FlatLand intends to simulate a natural world featuring an environment with limited resources, populated by simulated animals which require these resources to move around and create more of themselves.

FlatLand is a simple Multi Agent System. Put simply and in the context of computing, an agent is a piece of code which acts independently and makes decisions on behalf of some goal inside some sort of environment. If we consider a chat bot as an agent, then the chat room would be its environment.

FlatLands environment is a 2d grid of tiles, each with an energy level. FlatLands agents live on these tiles and need this energy to eat. They can see, move, eat, breed, and die. The purpose of FlatLand is to try and make a very simple simulation of the natural world. The environment is populated with some agents, who will move about and try to eat energy from the environment. When they have enough energy, they will try to breed with nearby agents of the same type as themselves. Some agents can also attain energy from other agents, simulating predation. The goal of an agent is to increase the population of its type. However, they have no awareness of this, they simply follow their programmed behaviour of looking for food and mates.

Agents possess 3 attributes: **Size**, **Creation** **Size**, and **Range**. Size is used for things like calculating how long agents live for, how much energy they need, or if they can or can’t be predated on by others. Creation Size controls how many children an agent will have, this ranges from 1 to 8. The more children an agent has the less energy each child starts with. Range controls how many tiles the agent can move in one turn, moving more tiles costs more energy.

When a child agent is born, one of these 3 attributes may mutate. Whose parent attributes it inherits are decided randomly. Then there is a chance of these values will increase or decrease. This system can be used to simulate natural selection by means of inheritance and mutation.

As of now FlatLand has no practical use outside of personal exploration.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# 1 The System

## 1.1 The Simulation

Below is an image of the simulation view. Here the environment is coloured dark grey, the less energy a tile has, the darker it is. You can see small patterns and shapes all over the simulation, these are tiles with less energy. An agent has visited them and eaten some of the energy. If you look closely, you may see some trails left behind agents as they move and eat.

Background pattern

Description automatically generated

You can see agents spread out all over the environment, however there are a few spots with a lot of the same colour. These are population explosions, enough agents of the same type met in a location with a lot of energy, causing a population boom. You can see the tiles towards the centre are considerably darker, showing how the population eats all the energy in one place before spreading out.

The below image is the simulation controls.

Graphical user interface, text

Description automatically generated with medium confidence

**Run Step** runs the simulation for one step. During one step every agent will have a turn and each tile has a chance of regaining energy.

**Start** simply runs the simulation indefinitely.

**Run For** lets you choose how many steps to run the simulation for.

**Populate** lets you place agents on the environment. The value below is the percent of tiles an agent should be placed on, i.e. at 100 an agent will be placed at every tile.

**Replenish Energy** resets the environments energy to its maximum amount.

**Clear** removes all agents from the simulation and resets the steps to 0.

The below image is the simulation controls when the simulation is running. You can only replenish the environments energy when the simulation is running.

Graphical user interface

Description automatically generated with low confidence

## 1.2 Environment Settings

The below image is the environment settings, you can update aspects of the environment from here.

**Max Energy** is the maximum amount of energy each tile can store.

**Min Energy** is the minimum amount of energy each tile can store.

**Regen Chance** is the percent chance a tile will regain energy each step.

**Regen Amount** is how much energy each tile will regenerate by.

**Environment size** is how many tiles one edge of the environment has. This must be between 1 and 600 and must also be a factor of 600.

Graphical user interface, application

Description automatically generated

The 7 buttons below control the environment colours.

**Terrain Colour** controls what colour terrain tiles are displayed as, while the rest control what an unoccupied tile looks like at the specified energy level.

All environment settings bar environment size can be changed while the simulation is running.

## 1.3 Agent Settings

This is the agent settings section. Here you can select an agents name and colour while seeing its ID. You can also change an agents **Size**, **Creation** **Size**, and **Range** here. You can see the resultant calculated attributes below. This is where you select how likely it is an agents child will mutate at birth, and the weight applied to this agent when the populate button is clicked.

Graphical user interface

Description automatically generated

If you select the **Motivations** tab, you will be shown the below view:

Graphical user interface

Description automatically generated

Here you can select if an agent gets its energy from the environment (Grazer) or from other agents (Predator) or both. You may also select a bias and a weight for each motivation. The bias is an additive value and can be used to add a small bias between motivations, while the weight is multiplicative.

If you then select the **Colour Model** tab, you will be shown the following view:

Waterfall chart

Description automatically generated with medium confidence

Here you can select the colour model used by the agent. **Static** means its colour never changes. **Attributes** means its colour is generated from the chosen colour and the agents attributes. It can be used to observe mutations as the simulation runs. **Random** will randomly add the random magnitude value to each colour channel every time an agent is born. This can be used to track different populations of the same type. It is useful in cavernous environments where populations can branch down multiple routes.

## 1.5 Diagnostics

The diagnostics view simply allows you to view useful simulation information that can’t be displayed in the simulation view. The **Info Log** displays useful messages.

Graphical user interface

Description automatically generated

Setting **Info Log Verbosity** to high will cause the system to log a message for every agent born.

## 1.5 Terrain Generation

You can access terrain generation from the systems tool bar:

Graphical user interface, text, application, chat or text message

Description automatically generated

**Generate Cave** paints a cave, while **Generate** **Cave** **System** will generate a system of interconnected branching caves.

**Fill** **Terrain** and **Clear** **Terrain** will set all tiles to terrain or clear all tiles of being terrain respectively.

Terrain Settings opens the terrain settings menu:

Graphical user interface, application

Description automatically generated

The checkbox **Fill** sets whether the shapes are painted as terrain, or the absence of terrain.

At the base level, terrain is generated by painting circles. Circles are then painted in clusters of which the density can be controlled. At a low density, a sparse cluster of circles is painted. At a high density, rough circle like shapes are generated. **Rock** **Size** determines how large the base circles are. **Cave** **Density** determines how dense the clusters should be.

To generate a cave, these clusters are painted in a line with the direction randomly changing. The chance of the direction changing is controlled by **Bend Density**. As the clusters are painted, their size will oscillate between **Upper** **Cave** **Size** and **Lower** **Cave** **Size**. Cave Wave can be considered the frequency of the wave but inverted. The chance of a cluster being placed corresponds to **Cave** **Weight**. The amount of caves that will be placed and their length corresponds to **Cave** **No** and **Cave Length** respectively.

Caves within cave systems are connected using fixed density clusters. The size of which is controlled by **Cavern** **Size**.

## 1.6 System Controls



The system controls can be accessed from the tool bar. Here you can save/load agents, the environment, or the entire simulation. Access pre-set simulations and toggle the controls.

# 2 Known Issues

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## 2.1 Error: could not open ‘%FLATLAND\_LOCATION%\jre\lib\jvm.cfg’

Graphical user interface, text, application

Description automatically generated

When **FlatLand1.0a.exe** is executed this window may be shown. Click OK to proceed.

## 2.1 Controls do not respond.

During runtime, it has been noted that the simulation controls occasionally stop responding to user input. The workaround for this issue is to restart the software.