
University of Glasgow | School of Computing Science

# Blobs on a Plane

George Popa

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Level 4 Project — February 28, 2016

**Abstract**

The project involves building an evolutionary simulator, dealing with observations of mimicked natural evolutions. It is intended as a teaching aid, with the user being able to intervene and change parameters in the simulation. Creatures called ”blobs” evolve and adapt to the environment, allowing the user to study group behaviours and population dynamics.

A on-line deployment is available: `http://unityblob.s3-website-eu-west-1.amazonaws.com`

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

While multiple systems implementing genetic algorithms are available (2.2), they tend to be obsolete, no longer supported, or generally intended as frameworks providing only the algorithms to be used in optimization. The project is intended as a light-weight educational utility, in order to provide insight into the dynamics of an evolving population under environmental conditions. The purpose of the simulation is to observe how average values of parameters change with respect to changes in the environment.

## 1.2 Overview

## 1.3 Dissertation Outline

The rest of the dissertation presents in detail the process of development of the evolutionary system, comically named “Blobs on a Plane”. The outline is as follows:

Chapter 2 deals with some of the previous research carried out in the field, as well as other pre-existing applications.

Chapter 3 details the functional and non-functional requirements, as well as the requirements gathering process.

Chapter 4 discusses various concerns about the design of the system and the user interface, any assumptions, and the design decisions made. Presented are also the initial design ideas and the final design.

Chapter 5 describes the implementation details, choice of framework, some of the evolutionary algorithms used, as well as the project deployment.

Chapter 6 deals primarily with evaluation, from both users and experts, as well as internal testing.

Chapter 7 provides a summary of the project, project management matters, and future work.

# Chapter 2

# Background

This chapter aims to present the reader with useful explanations of the field-specific terms used in the dissertation as well as provide insight into the significant research previously carried out. Included are example of pre-existing applications, either implementing similar algorithms, or possessing a comparable purpose.

## 2.1 Definitions

In order to provide a better understanding, some terms are defined below:

- "Blob" - A single creature in the simulation;
- Food - A nutritional pellet, providing Energy to whomever eats it;
- Population - The set containing all "blobs: at a certain time;
- Energy - A "blob's" measure of how healthy he is. This triggers death at low values;
- Reproduction - The act of division of a "blob" when an energy threshold is reached;
- Patience - The time measured in number of ticks until a "blob" leaves a certain area.
- Genome - A simulated DNA string in which the parameters of a "blob" are encoded.
- Mutation - The act of a single bit in the genome becoming inverted at random.
- Crossover - The act of combining two genomes at a random point, in order to generate two new genomes.

## 2.2 Previous Work

Evolutionary computing is the field of artificial intelligence devoted to solving problems by implementing Darwinian principles of evolution. Starting from the premise of a common biological ancestor, Darwin proposed evolution from the simple to the complex, given enough time. With mutations appearing in a creature's genetic code, he stipulated that beneficial mutations providing an advantage would be passed on to future generations.[3] Accumulating multiple such mutations would lead to a completely new organism, different from the original.

### 2.2.1 Evolutionary Algorithms

When faced with complex optimization problems, performing a linear search, especially through a large solution space usually associated with such problems, may prove ineffective.

While initially proposed by Alan Turing in 1950, evolutionary algorithms became regarded as a method of solving complex optimisation problems in the 1970s. In general, several biological methods, such as sexual or asexual reproduction or genetic mutation, are used in order to evolve the population of a particular system with respect to a certain predetermined fitness function. As such, more "fit" individuals have a higher chance of being selected to pass on their genome to the next generation, thus increasing the fitness factor of the overall population after each iteration. Over a longer period of time spanning from a few thousand, to a million generations, the population will evolve towards a solution. This, however, suffers from diminishing returns [4], as the population tends to meet an impasse in locally optimal points, rather than reach the global maximum.

### 2.2.2 Biological Simulation

A more prominent paper in the field of artificial evolution is "Evolving Virtual Creatures" by Karl Sims [11]. The paper details an evolutionary system which focuses evolving a population of creatures consisting of three dimensional blocks with respect to various movement-specific tasks (such as walking, running, jumping, climbing, and swimming). Noteworthy is the fact that both the creature's morphology and it's "brain" are evolved. While

Figure 2.1: A screenshot from the simulator build by Karl Sims "Evolved Virtual Creatures"

the initial randomly created generation was incapable of performing any task, subsequent generations, showed significant improvement. Even after a relatively low number of generations, some creatures were becoming adapted to their environment. In the swimming example, some creatures were successful in developing fin-like appendages, used for locomotion, while other replicated the waving motion of water-snakes. The scope of the experiment, however is quite limited, as creatures are observed individually, with a rather basic fitness function, and not in competition with each other.

A more recent paper, Dan Lessin's et al. "Open-ended behavioral complexity for evolved virtual creatures." [7], deals with the open-ended nature of evolution, previously unexplored in Karl Sims's paper. The paper proposes the evolution of a creature's brains as well as body, thus allowing it to better adapt to its environment. While the evolution of the body is fairly similar to the method described by Karl Sims, the control part of a creature is evolved differently. This involves encapsulating a creature's previously learned skills. The benefits presented are persistent memory, allowing a creature to store a certain behaviour once it has been learned, and overall simplicity, allowing the whole procedure to be called as a single node, instead of actively recreating the movement. By allowing newer nodes to be modified more frequently than older ones, the algorithm simulates

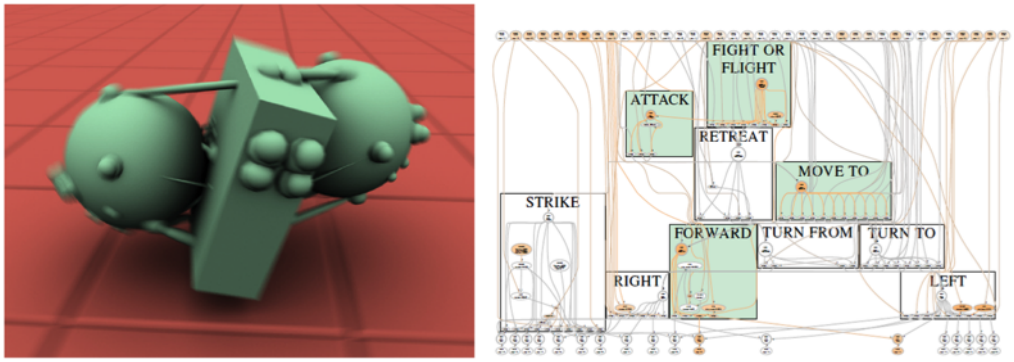


Figure 2.2: An example of a creature's physiognomy and control circuit

the forming of habits. By changing the fitness function, creatures can adapt to different environments, or learn more complex behaviours, without "forgetting" their previous training. This, in turn, provides a more accurate representation of nature.

### 2.2.3 Virtual Worlds

While research in the field of Genetic Algorithms is extensive, it deals mostly either with problems of optimization, or with the best few individuals of a population. This is not the case when trying to simulate the entire population, as being the very best only gives a marginally higher chance of passing genes to the next generation. Simply being "good enough" to eat sufficient food in order to be able to live and reproduce is adequate. This is especially important when dealing with a heterogeneous environment, where food density might vary. In this scenario, simply being in an area with more food could be considered more beneficial than becoming the best individual. A common occurrence in such simulations is the gathering of individuals around food clusters, with those to do so dying off.

#### Darwinbots

A similar application is appearance is Darwinbots [2]. Darwinbots presents itself as an artificial life simulator, where creatures, called "bots" can compete for food in a simulated two dimensional environment. The most successful "bots" will live on to pass their genes to subsequent generations. Darwinbots, however, specialises in simulating Von Neumann machines.

Darwinbots specialises in providing a platform for interactivity between multiple species of organisms with wildly varying behavioural patterns. Should it prove beneficial to do so, creatures may combine different purposes to form more complex, specialised, multi-cellular organisms, with constituent parts focusing independently on locomotion, sensory input, digestion, and combat.

The creatures implement a piece of code represented as a syntax tree as their inherent algorithm. Using tree cross-over operations described in "A Field Guide to Genetic Programming" [10] these algorithms, which essentially form the control circuit of a particular individual, are evolved. Due to the nature of mutation in tree cross-over individuals of the same species can evolve in different directions, should both be more optimal than the current incarnation, due to divergent evolution.

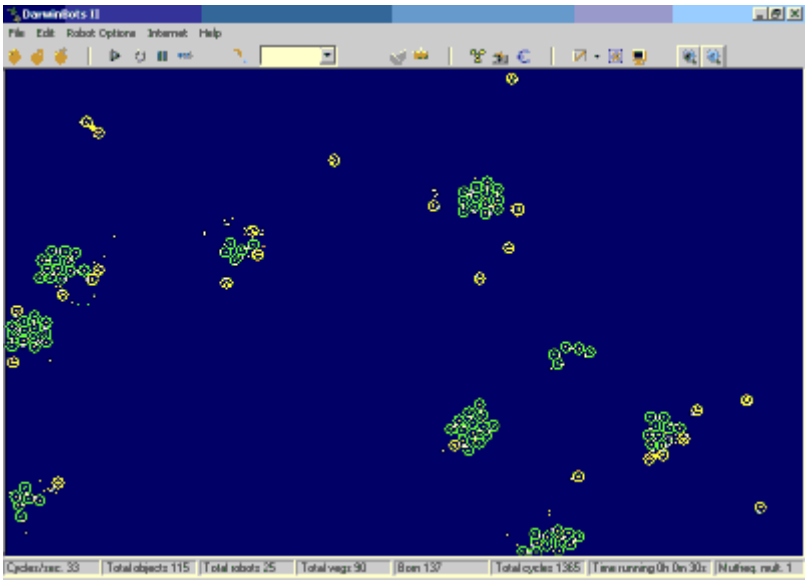


Figure 2.3: A screenshot from a Darwinbots simulation

### Goopies

A more simple approach is taken by Paul T. Oliver in his project called “Goopies”.[9] The creatures live in a bounded, circular world, together with food pellets and obstacles. A creature’s behaviour is determined by its intrinsic neural network. Creature physiognomy exists only for the purpose of conveying information about certain parameters of its control circuit. Unlike Darwinbots, at the end of one generation, the best individuals with respect to a fitness function are selected. Crossover is performed on their genes and a new population is created.

In this simulate world, creatures do not actively interact with each other like in Darwinbots. The impact made by an individual on the others is the food he consumes and his corpse. This has generated a peculiar behaviour where “Goopies” would linger close to others close to their death, similar to carrion-eater behaviour in animals.

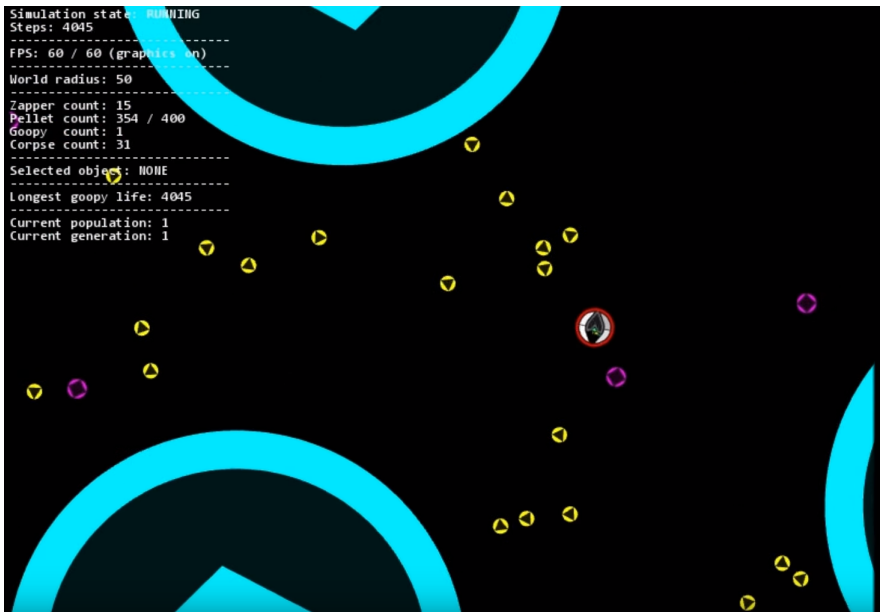


Figure 2.4: A “goopie” searching for food

**Conway's Game of Life**

Introduced by John Conway in 1970 [5], the eponymous game is a two-dimensional implementation of a cellular automation. According to a predefined rule, the subsequent states are created from an initial state. Building on von Neumann ideas of a machine capable of self-replication, Conway attempted to simplify the concepts. Due to the fact that it implemented a universal Turing Machine [1], this sparked interest for the game since its inception.

Figure 2.5: A cellular formation that has the property to generate "gliders"

It is particularly simple to observe patterns among the population. Notable among the categories of patterns are "still lifes", "oscillators", and "spaceships". The first denote a series of arrangements of cells that, if left alone, do not change in any way.[6]. Oscillators are a set of repeating patterns, characterised by a period. "Spaceships" are patterns that can translate themselves across the grid. Two gliders can be shot at a particular angle at a static two-by-two block in order to move it. This can act as memory for a more complex system implementing a finite-state machine.

# Chapter 3

# Requirements

## 3.1 Requirements Gathering Process

## 3.2 Functional / Non-Functional

### 3.2.1 Functional Requirements

### 3.2.2 Non-Functional Requirements

# Chapter 4

# Design

The system simulates a population of individuals, named “blobs”, whose sole purpose is reproduction. This is done asexually, when each individual reaches a high enough energy threshold, encoded in his genome. Energy can be gained by eating the food pellets scattered randomly around the world. The “blobs” eat food by colliding with it, and find it by performing a random walk, in this case, a simplified version of a Lévy walk[8]. The system is made with interactivity in mind, allowing the user to tweak various parameters of the environment, while observing the population evolves in order to adapt to the change. A user could interact more directly with the simulation by adding “blobs” or placing food. A sudden influx in either will generate a period of instability in the population which eventually reverts back to a stable state. Information about the population is presented via a graph, showing total number, and average characteristics of the individuals. More specific information about a “blob” in particular is displayed via a pop-up message, should a creature be clicked on.

## 4.1 Goals and Considerations

## 4.2 User Interface

### 4.2.1 User Controls

### 4.2.2 Data Interface

## 4.3 Application Design

# Chapter 5

# Implementation

## 5.1 Framework Choice

## 5.2 Overall Architecture

## 5.3 Main Logic

### 5.3.1 Simulator State

### 5.3.2 Update

### 5.3.3 Save

## 5.4 Blob Logic

### 5.4.1 Characteristics

### 5.4.2 DNA

### 5.4.3 Update

### 5.4.4 Movement

### 5.4.5 Reproduction

## 5.5 Deployment

# Chapter 6

# Evaluation

## 6.1 Testing

## 6.2 User Evaluation

### 6.2.1 Feedback Forms

### 6.2.2 Expert Opinion

## 6.3 Feedback Integration

# Chapter 7

# Conclusion

## 7.1 Summary

## 7.2 Project Management

## 7.3 Future Work

## 7.4 Reflection

# Bibliography


[1] Life universal computer. http://www.igblan.free-online.co.uk/igblan/ca/index.html.

[2] Darwinbots. http://wiki.darwinbots.com/w/Introduction, 2005.

[3] Charles Darwin.

[4] Maria R Fumagalli, Matteo Osella, Philippe Thomen, Francois Heslot, and Marco Cosentino Lagomarsino. Speed of evolution in large asexual populations with diminishing returns. *Journal of theoretical biology*, 365:23–31, 2015.

[5] Martin Gardner. Mathematical games: The fantastic combinations of john conways new solitaire game life. *Scientific American*, 223(4):120–123, 1970.

[6] David Griffeath. *New constructions in cellular automata*. Oxford University Press, 2003.

[7] Dan Lessin, Don Fussell, and Risto Miikkulainen. Open-ended behavioral complexity for evolved virtual creatures. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 335–342. ACM, 2013.

[8] Franziska Matthäus, Marko Jagodič, and Jure Dobnikar. E. coli superdiffusion and chemotaxissearch strategy, precision, and motility. *Biophysical journal*, 97(4):946–957, 2009.

[9] Paul T Oliver. Goopies. http://forum.codecall.net/topic/72637-goopies-evolving-neural-networks-wip/?p=643047, 2012.

[10] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A field guide to genetic programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008.

[11] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.