Parallelising Swarm Chemistry

Robert Stamper—Supervised by Seth Bullock

Motivation

Life is full of systems that demonstrate complex macroscopic structure as a result of the behavior of many independent interacting agents (ants, termites, bees, human tissue growth)

We lack a deep understanding of the links between the microscopic behavior of the individual agents maps to the macroscopic structure.

The emerging field of morphogenetic engineering is exploring the parallels between naturally evolved and artificially engineered systems with the aim of discovering underlying principles.

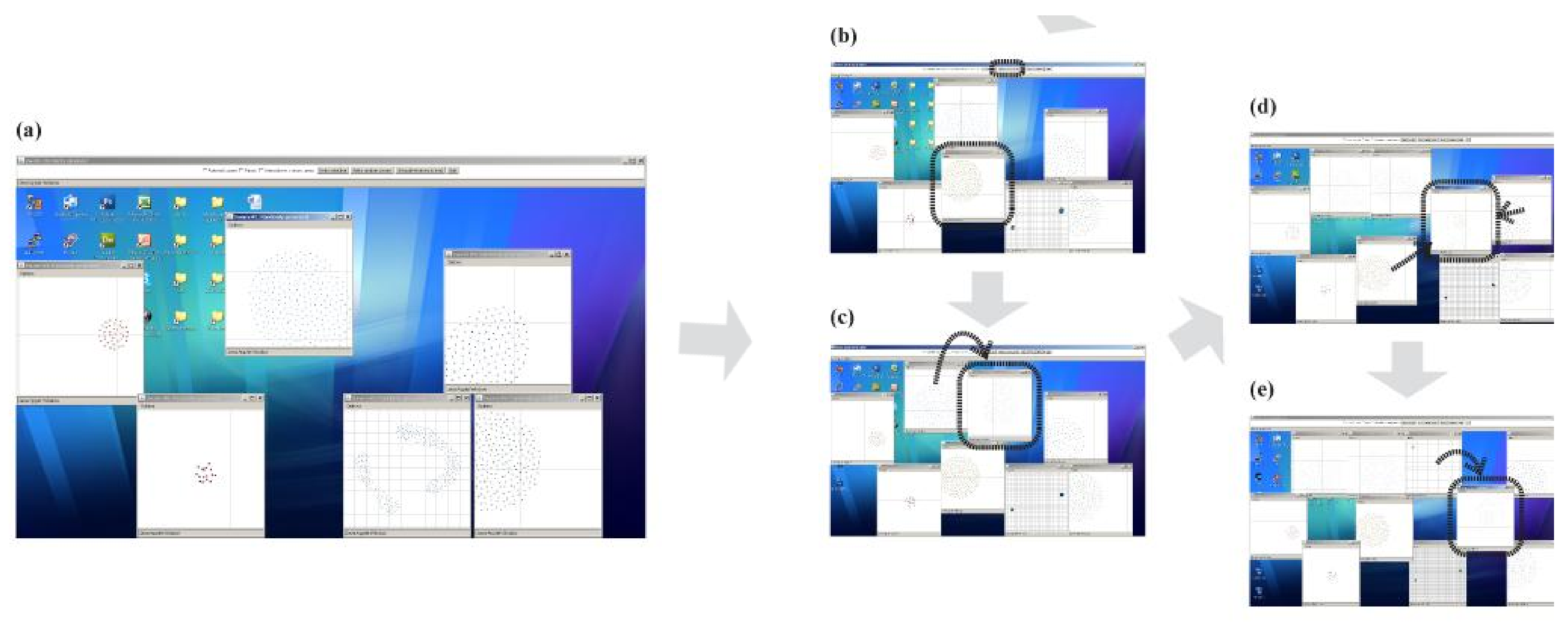
One example of a simple complex system capable of demonstrating complex structure is Hiroki Sayama's Swarm Chemistry model. Swarm chemistry combines the movement model of Reynold's Boids with the notions of 'competition functions' and genetic mutations. Reynolds came up with his movement rules after studying the movements of flocks of birds and fish.

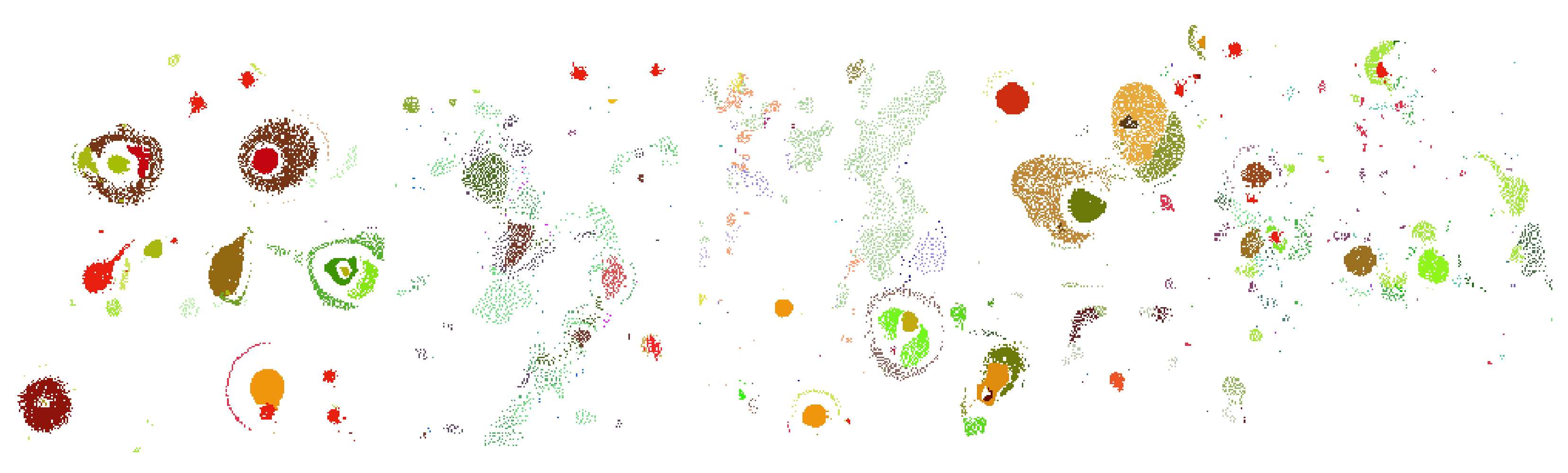
The original simulator

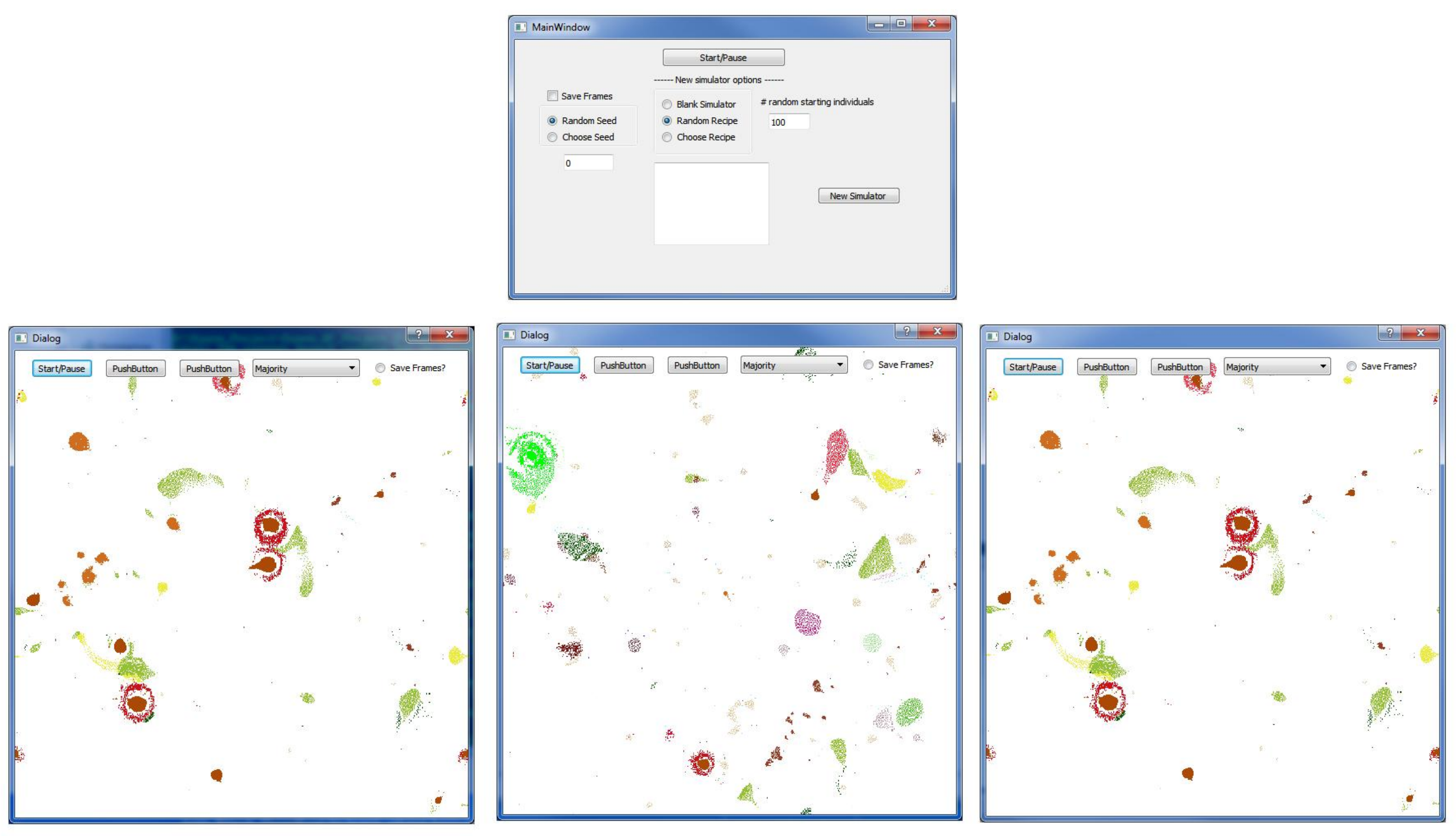
Sayama quickly prototyped his swarm chemistry model in Java and then moved on. Swarm Chemistry has been shown little love in recent years and the old simulator is very rough to use, and runs serially on one CPU core.

Because the space of possible movement rules the particles can have is so large, and what passes for an "interesting" swarm is hard to quantify, Sayama took an interactive evolutionary approach to finding interesting swarms.

The original simulator allowed the user to create multiple populations and let them evolve in parallel. The user could mix interesting swarms with each other (kind of like an "island model" of evolution).







[✓ Random Seed] Choose Seed: 0

[✓ Random Recipe] # random starting individuals: 100

Aims

To create an efficient new simulator in C++ which makes use of multiple CPU cores or, if available, a GPU

To implement new features which make exploring the parameter space easier

A modular code base which facilitates easier experimentation.

To potentially demonstrate some form of extension of the SwarmChemistry model (new competition functions or environmental perturbations for example)

Technologies/frameworks

The simulator is implemented in C++11 using the QT library and OpenCL

Challenges

Programming GPUs is generally a pain

Things like sorting and generating random numbers are awkward

Balancing between usability and performance

Implemented a sound system from a software development point of view that could be easily understood by people wanting to extend it and implement modified or new models

Future work

Investigate if automated evolutionary methods using quantitative metrics can be used to look for "interesting" swarms