Evolution of Multi-Agent Systems

by multi-type, self-adaptive genetic programming

Taskable Agent Software Kit Principal Investigators' Meeting
February 19-20, 2003

Lee Spector
Hampshire College
lspector@hampshire.edu, http://hampshire.edu/lspector

Outline

- Guiding questions and quadchart
- Push/Breve/SwarmEvolve updates/integration
- Goal/target dynamics
- New Diversity Metrics
- Emergence of collective organization (x2)
- Integration of Elementary Adaptive Modules

Guiding Questions

- Can evolution be used to help discover effective *controllers* for multi-agent systems?
- Can evolution be used to help discover *design principles* for multi-agent systems?
- Can evolution be used to help *analyze controllers/principles* for multi-agent systems?
- Can we provide general evolution-based software for a Taskable Agent Software Kit?

Evolution of Multi-Agent Systems

by multi-type, self-adaptive genetic programming

Hampshire College

Design Problem/Solution Approach	Experiment/Analysis Methodology
• Mobile, potentially evasive goals	• Baseline: infinite stability or goal random walk
• Self-adaptive evolution of agent controllers	• Baseline: infinite reward per goal
• Push language for evolved agent programs	• Baseline: multiple input gradient descent
• Breve 3D/physical simulation environment	• Baseline: no communication/coordination
Metrics	**Results**
• Coverage (depletion of "food" supply)	• General purpose agent evolution software
• Response delay for categorical changes	• Emergence of collective behavior (mult. forms)
• Individual lifetimes, parsimony, diversity	• Analysis of stability/adaptation interactions
• Information and energy sharing	• New diversity metrics

Push

- Stack-based language with one stack per type; types include integer, float, vector, Boolean, code, child, type, name.
- Evolved agents may use:
  - multiple data types
  - subroutines (any architecture)
  - recursion
  - evolved control structures
  - evolved evolutionary mechanisms

Breve

- Written by Jon Klein, http://www.spiderland.org/breve
- Simplifies rapid construction of complex 3D simulations.
- Object-oriented scripting language with rich pre-defined class hierarchy
- OpenGL 3D graphics with lighting, shadows, and reflection.
- Rigid body simulation, collision detection/response, articulated body simulation.
- Runge-Kutta 4th order integrator or Runge-Kutta-Fehlman integrator with adaptive step-size control.

Autoconstructive Evolution

- The ways in which programs reproduce and diversify are themselves products of evolution.
- Agents control their own mutation rates, sexuality, and reproductive timing.

Push/Breve Integration

- C-language Push interpreter plugin (original was Lisp, Java versions by others).
- Push interpreter per Breve agent.
- Breve agents can perform/evolve arbitrary computations.
- Push/Breve callbacks implement sensors/effectors.
- XML specification for Push standardization.

SwarmEvolve 1.0-1.5

- Acceleration = p1*[away from crowding others vector]
  + p2*[to world center vector]
  + p3*[average neighbor velocity vector]
  + p4*[to neighbor center vector]
  + p5*[random vector]
  + p6*[away from other species vector]
  + p7*[to closest energy source vector]
- Genotype = [p1, p2, p3, p4, p5, p6, p7].
- Various energy costs (collisions, species outnumbered, etc.).
- Upon death (energy = 0), parameters replaced with mutated version of fittest (max age * energy) of species.

SwarmEvolve 1.5

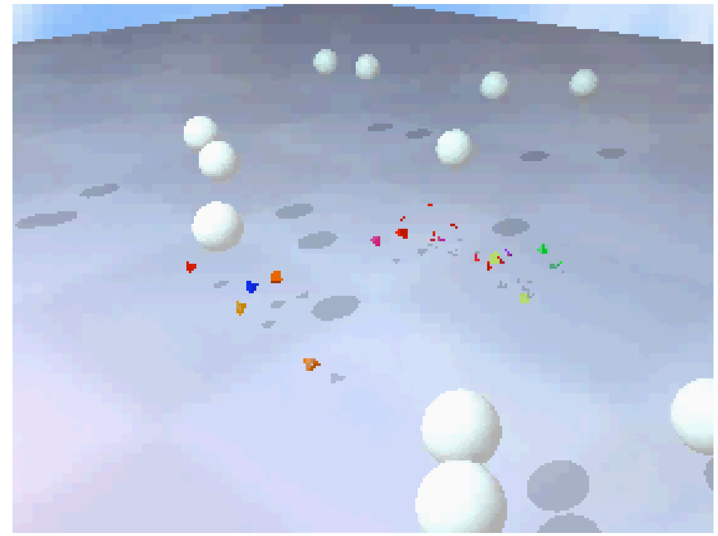
- Food consumption/growth, redesigned feeders (goals/targets).
- Birth near mothers.
- Corpses.
- Food sensor, inverse square signal strength.
- GUI controls and metrics.

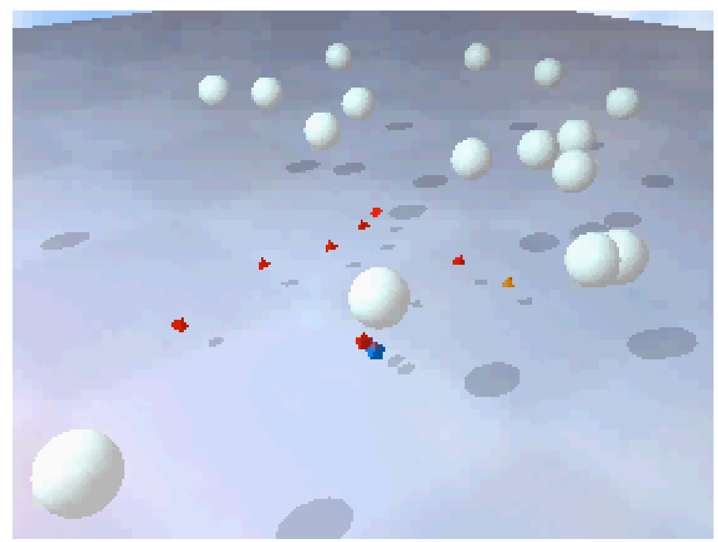
SwarmEvolve 2.0

- Behavior (including reproduction) controlled by evolved Push programs.
- No hard-coded species. Color, color-based agent discrimination controlled by agents.
- Energy conservation.
- Facilities for communication, energy sharing.
- Enhanced user feedback (e.g. diversity metrics, agent energy determines size).

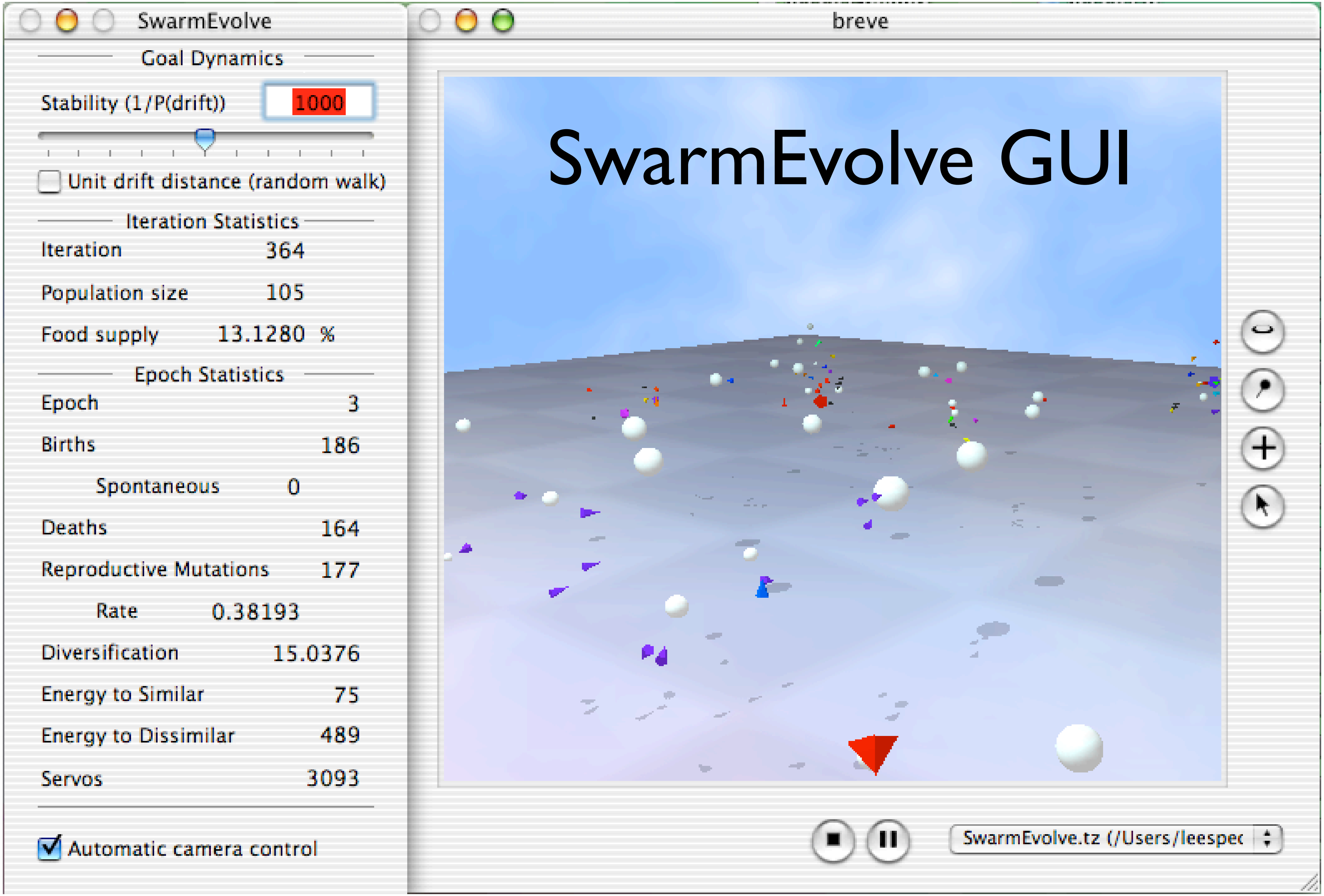
Goal/Target Dynamics

- Implemented:
  - Linear drift
  - Random walk
- Planned:
  - Evasive
  - Flocking
- Many parameters





Potentially: integrate with Alphatech problem generator/TTAS


SwarmEvolve GUI
SwarmEvolve
breve
Goal Dynamics
Stability (1/P(drift))
1000
Unit drift distance (random walk)
Iteration Statistics
Iteration 364
Population size 105
Food supply 13.1280 %
Epoch Statistics
Epoch 3
Births 186
Spontaneous 0
Deaths 164
Reproductive Mutations 177
Rate 0.38193
Diversification 15.0376
Energy to Similar 75
Energy to Dissimilar 489
Servos 3093
Automatic camera control
SwarmEvolve.tz (/Users/leespec

New Diversity Metrics

$$diversity(P) = \frac{\sum_{i \in P} \frac{|\{j \in P : \Delta(i,j) > \delta\}|}{|P|-1}}{|P|}$$

- Average, over all agents, of proportion of remaining population considered “other”
- Genotypic (e.g. code, code size)
- Phenotypic (e.g. color, behavior, signals)
- Reproductive/developmental

Collective Organization

- Multicellularity in SwarmEvolve 1.0.
- Energy sharing in SwarmEvolve 2.0.

Multicellularity

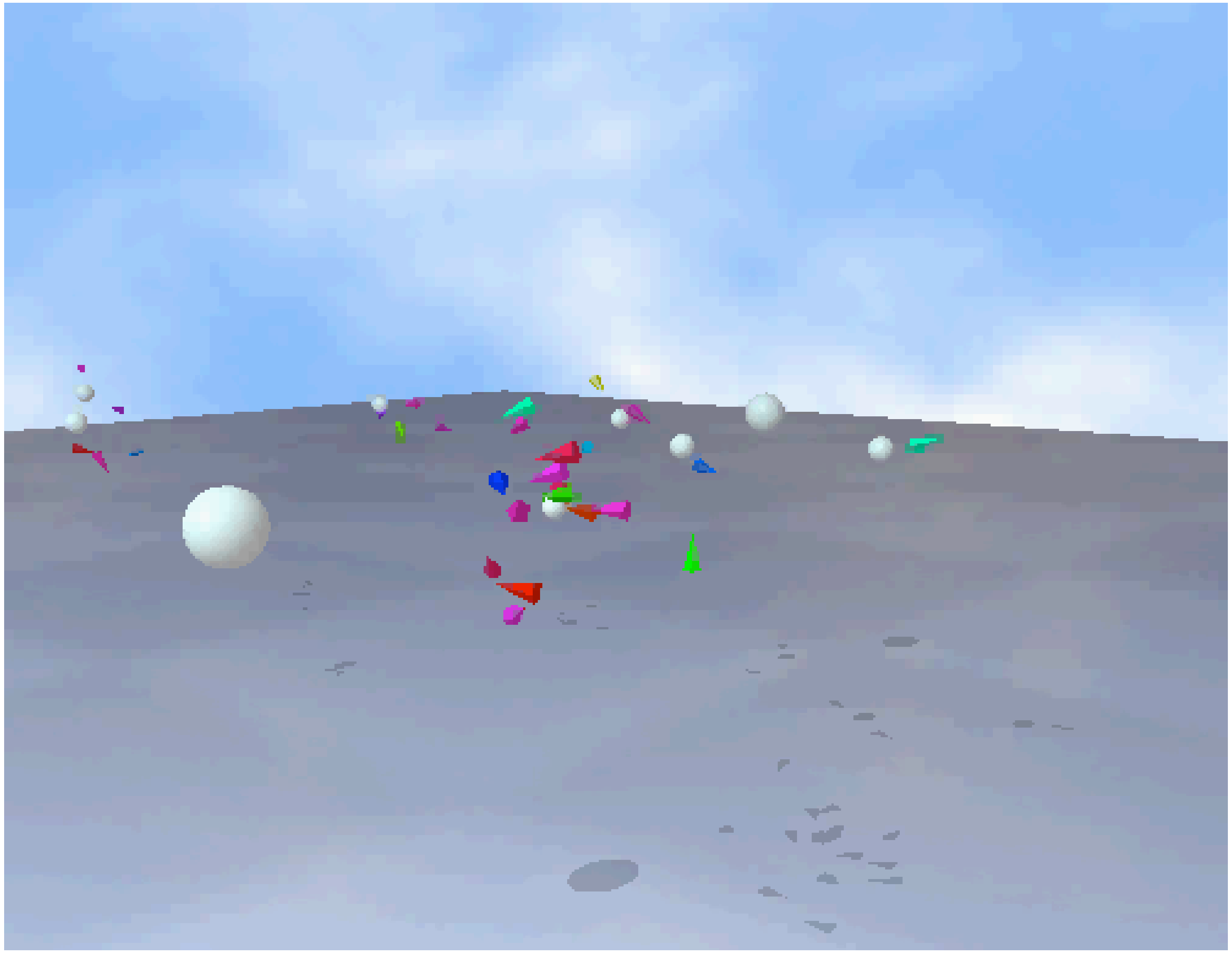
- Observed behavior: a cloud of agents hovers around an energy source. Only the central agents feed, while the others are continually dying and being reborn.
- Can be viewed as a form of emergent collective organization or multicellularity
- Peripheral agents: defensive organs.
- Central agents: digestive/reproductive organs.

Multicellularity

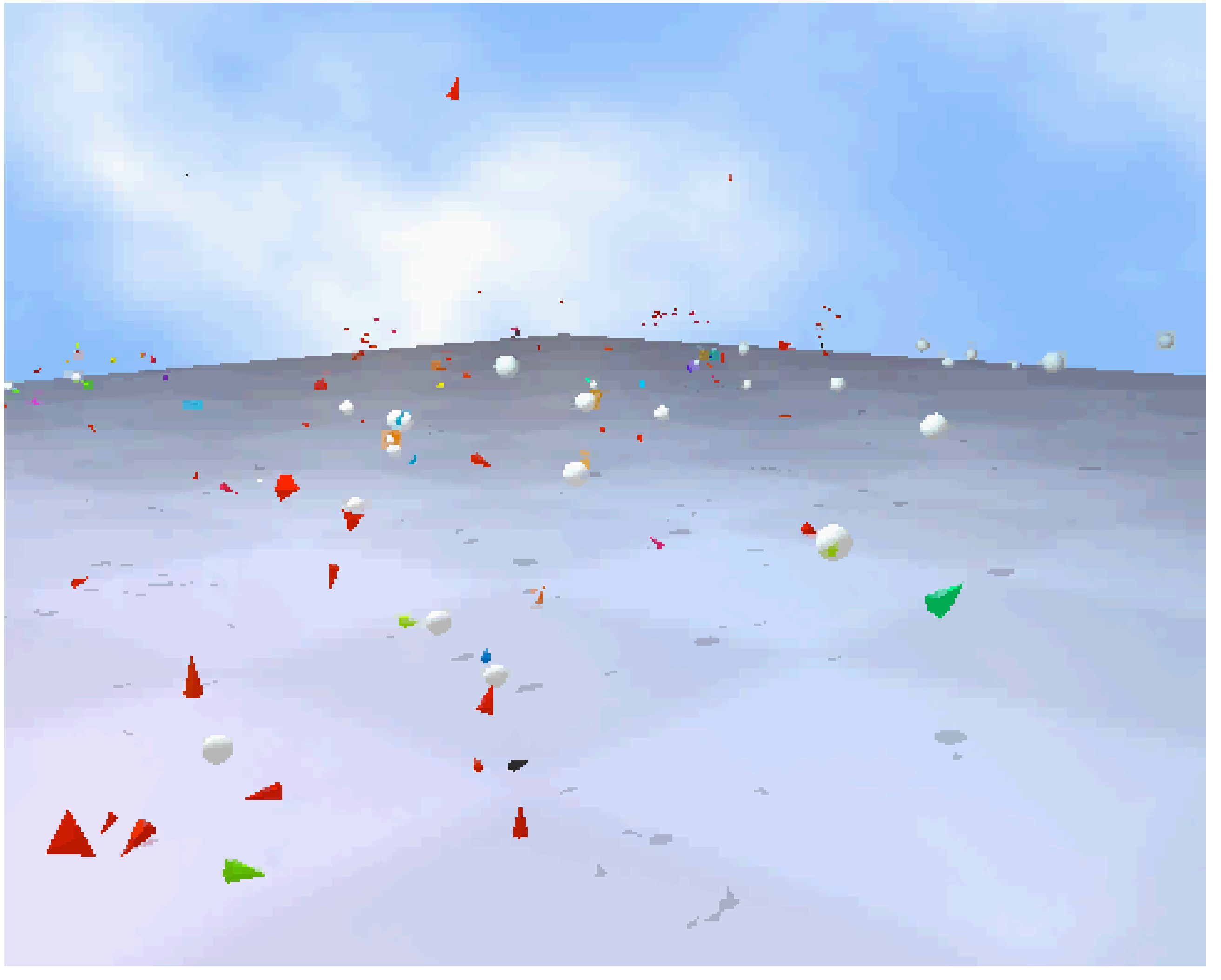
Energy Sharing

- Example evolved strategy: Reckless goal-seeking + sharing.
- Functional instructions of evolved code: `( toFood feedOther myAge spawn randF )`
- Accelerates directly toward nearest goal, feeds others, and turns random colors.
- Evolved mutation regime: rate $\propto$ 1/age.
- High goal coverage, low lifetimes.

Energy Sharing



Other Strategies



Servo/EAM Integration

- Now: single EAM per agent.
  Potentially: any number, any architecture.
- Now: servo EAM only.
  Potentially: all EAM types.
- New Push instructions: setServoSetpoint, setServoGain, servo.
- Initial indications: high utility.

Evolved Servo Use

Example:

```
( spawn ( ( ( V* VECTORX myLocation (
CODECHILD ( servo OR ) ( setServoGain
mutate ) ) ) * ( DUP MAKEVECTOR (
foodIntensity ) ) ) ( CONS ( myHue (
crossover FALSE ) ( OR ) (
setServoSetpoint otherProgram friendHue )
) ) + ( QUOTE ) ) ( VECTORX V* ) ( (
NULL NTH ( AND ) ) ) ( CODECHILD randF
( V- foodIntensity ) VECTORZ ) ( DO*
FALSE ( myAge crossover NOOP ( feedFriend
) ) ( V* ( NULL ) ) ) )
```

OEF Correspondence

- Target dynamics ↔ energy source dynamics.
- Several OEF metrics apply (e.g. task service rates, delays).
- Some divergences incidental (e.g. floating targets, specific vehicle control parameters).
- Some divergences necessary (e.g. no evolution without death) but many "lessons learned" should generalize.

Group Linkages

- MIT/BBN re: Elementary Adaptive Modules.
- UNM re: modularity and evolvability.
- UMass re: mining of SwarmEvolve data.