Spatial Computing & the challenge of Engineered Emergence

Jacob Beal (& Jonathan Bachrach) MIT CSAIL

Networked devices are **filling** our environment...



Networked devices are **filling** our environment...



Networked devices are **filling** our environment...



How do we program aggregates robustly? 4

Outline

- What is Spatial Computing?
- Continuous Space Programs
- Proto & Amorphous Medium
- Engineered Emergence

Spatial Computers



Interior tre 67m tall

More formally...

- A spatial computer is a collection of computational devices distributed through a physical space in which:
 - the difficulty of moving information between any two devices is strongly dependent on the distance between them, and
 - the "functional goals" of the system are generally defined in terms of the system's spatial structure



More formally...

- A spatial computer is a collection of computational devices distributed through a physical space in which:
 - the difficulty of moving information between any two devices is strongly dependent on the distance between them, and
 - the "functional goals" of the system are generally defined in terms of the system's spatial structure









Example App: Mobile Streaming



Definition

































Why use continuous space?

- Simplicity
- Scaling & Portability
- Robustness

(we'll come back to this in a bit...)



Amorphous Medium



Continuous space & timeInfinite number of devicesSee neighbors' past state

Continuous



Approximate with:Discrete network of devicesSignals transmit state

Proto



21

Computing with fields



Computing with fields



Proto





Modulation by Restriction



In simulation...



Why use continuous space?

- Simplicity
- Scaling & Portability
- Robustness



Other Approaches

- Lots of C hacking...
- Regiment / WaveScope
- Viral: TOTA / Smart Messages / Paintable Computing
- Pattern formation: [Coore] / [Nagpal] / [Kondacs] / [Stoy] / [Goldstein], ...
- Abstract: Kairos, EgoSpaces, Logical Neighborhoods, "views", ...



On to Emergence...

- What is emergence?
 - Greater than the sum of its parts?
 - Unpredictable from local interactions?
 - Only definable for the aggregate?



"Engineered Emergence"

Routine design of the behavior of aggregates of unreliable devices with complicated interaction patterns.



"Engineered Emergence"

Routine design of the behavior of aggregates of unreliable devices with complicated interaction patterns.



Not quite...

• Nature's toolkit is still better

Continuous

Definition

3.5 weeks



TADAM.

Emergence

33

Proto is good for library building: global-tolocal + modulation by restriction

Proto

1st: Flocking



13 lines of Proto









2nd: Self-Healing Gradients



14 lines of Proto









35

3nd: Dispersion













Together: Guided Flocking











Four Useful Principles

- Self-Scaling
- Sparseness
- Gradual degradation
- Failure simplification



Gradual Degradation

- Decoupling by low sensitivity to
 - Implementation details
 - Parameter values
 - Conditions of execution
- Use when you don't understand or can't control the environment.



Gradual Degradation: Implementation Details

• Plane wave at different resolutions:



Self-Scaling

- Decoupling through geometry:
 - specification of behavior (units)
 - implementation details (coordinate system)
- Use when you don't know the relationship between the behavior you want and the details of its implementation



Failure Simplification

- Decoupling by preferential selection of preferred failure type
- Use when you don't understand or can't prevent failures

We're used to preventing failures. What if we just manage their impact?



Self-Scaling, Failure Simplification: Neighborhood Ops



Sparseness

- Decoupling by making unwanted interactions rare.
- Use when devices need to make noninterfering decisions independently.

If at first you don't succeed, just try again.



Sparseness: Symmetry Breaking

• Temporary leadership via 1/f noise

Definition





Contributions

- Many networks are spatial computers
- Continuous-space geometric programs are simple, scalable, and robust
- Proto compiles global descriptions into approximate local implementations
- Emergent behaviors can be engineered with by modulating regions
- Four useful principles for engineering emergence