Programmable Matter

- A programmable material...
- ...with actuation and sensing...
- ...that can change its physical properties ...
- ...under software control...
- ...and in reaction to external stimuli

For movie see [www.cs.cmu.edu/~claytronics/movies/cardesign.wmv](http://www.cs.cmu.edu/~claytronics/movies/cardesign.wmv)

Challenge At End

- Or, unabashed request for help.

Basic Human Need
An Abstract Communication Device

Sense physical phenomena

Encode and transmit data

Reproduce physical phenomena

Communication: a CS point of view

Audio:
- Encoding: sound waves
  - Microphone
  - Speaker
- Encoding: mp3

Video:
- Encoding: light
  - Camera/CCD
  - Display
- Encoding: mpeg4

Can we do better?

- Audio: 3D model
  - Multiple cameras
  - ?
Can we do better?

- Audio: sound waves
  - Microphone
  - Speaker
- Video: light
  - Camera/CCD
  - Display
- Pario: 3D model
  - Multiple cameras
  - Claytronics

An Advertisement from 2012

For movie see www.cs.cmu.edu/~claytronics/movies/cardesign.wmv

What we need to do

And don’t forget ...

Today

And, of course, movement

Tomorrow
A Computer Architects View

• Human need:
  what we want to do

• Moore’s law:
  what we can do

Moore’s Law
Moore’s Law

Where are we in 50 years?

<table>
<thead>
<tr>
<th></th>
<th>1949</th>
<th>2003</th>
<th>2050</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Eniac</td>
<td>greeting card</td>
<td>Programmable matter</td>
</tr>
<tr>
<td>Cost</td>
<td>5M-23M (2002 $)</td>
<td>1$</td>
<td>1 millicent</td>
</tr>
<tr>
<td>Weight</td>
<td>30 tons</td>
<td>1 oz</td>
<td>20 μg</td>
</tr>
<tr>
<td>Volume</td>
<td>450 M³</td>
<td>1 cm³</td>
<td>1 nm³?? (1 μm³)</td>
</tr>
<tr>
<td>Power</td>
<td>200KW</td>
<td>20mW</td>
<td>2 attowatts</td>
</tr>
<tr>
<td>Cycle time</td>
<td>&gt;200μs</td>
<td>25ns</td>
<td>2 picosec</td>
</tr>
<tr>
<td>Storage</td>
<td>&lt;800B</td>
<td>4KB</td>
<td>16KB</td>
</tr>
</tbody>
</table>

Cogent arguments for both sooner and later exist.

Claytronics

- A programmable material...
- ...with actuation and sensing...
- ...that can morph into shapes under software control...
- ...and in reaction to external stimuli

Types of “Programmable Matter”

- Modular Robots
- Sensor Network
- Claytronics
- Synthetic Biology
- FPGAs
- Nanotechnology

For Movie see: [www.cs.cmu.edu/~claytronics/slice-of-cardesign.avi](http://www.cs.cmu.edu/~claytronics/slice-of-cardesign.avi)
Scaling

• Goal: Form dynamic high-fidelity macroscale objects

High-fidelity ⇒ sub-millimeter units
Macroscale ⇒ millions of units
Scaling: Down in size
Up in number

Method:
Ensemble Effect

Node Requirements

• Each node must have
  - Computation+Memory
  - Communication
  - Energy Storage
  - Sensing
• Yet, scaling demands nodes be:
  - Simple
  - Small
• Ensemble Axiom:
  A node should include only the functionality necessary to achieve the desired ensemble.

Challenges

• Challenges are all intertwined
  - Hardware/Software trade-offs
  - Ensemble Axiom
• Example: Energy
Challenges

• Challenges are all intertwined
  - Hardware/Software trade-offs
  - Ensemble Axiom
• Example: Energy

Node Ensemble

Hardware

Software

One Possible Path

Today

Tomorrow
Harness Photolithography

• Use Moore's law!
• Challenge:

How to create a 3D object from a 2D process?

• Start with standard Wafer
• Create our circuits in a special pattern

Reid, AFRL

300 microns
Harness Photolithography

• Use Moore’s law!
• Challenge:
  How to create a 3D object from a 2D process?

• Start with standard Wafer
• Create our circuits in a special pattern
• Post-process to create sphere

A sanity check

1 mm diameter sphere
Mass < 1 mg

Electrostatic Actuators
~5 body lengths / sec

Communication Capacitors

Power Storage
Supercapacitor stores enough energy to execute over 200 million instructions or move 2 million body lengths

Computation Capability
8086 Processor with 256KB memory
SOI-CMOS 90 nm process with > 2M transistors.

Power distribution
Transmission of “energy packets” using capacitive coupling fills reservoir in < 1μs.

Research Plan

• Investigating hardware
  - Mechanisms
  - Systems
• Investigating Software
  - Languages and models
  - Algorithms
  - Tools
Investigating many size scales

Stochastic Catoms
Magnetic Based Catoms

For movie see: [www.cs.cmu.edu/~claytronics/movies/3catoms.avi](http://www.cs.cmu.edu/~claytronics/movies/3catoms.avi)

Ensemble Effect & Motion

For movie see: [www.cs.cmu.edu/~claytronics/movies/es-tube.avi](http://www.cs.cmu.edu/~claytronics/movies/es-tube.avi)

Initial Experiments at <1mm-scale

For movie see: [www.cs.cmu.edu/~claytronics/movies/es-tube.avi](http://www.cs.cmu.edu/~claytronics/movies/es-tube.avi)

What about the software?

Today  Tomorrow
Two Broad Problem Areas

• Programming the Ensemble: How does one think about coordination of millions of elements?
• Programming the Unit: What is the programming model for a (single) element?

• Let’s focus on the ensemble

“Emergent Behavior”

• Incredibly seductive
• Witness ants

• They proceed from nest to food and back via pheromone trail

“Emergent Behavior”

• Incredibly seductive
• Witness ants

• What happens when it is blocked?

“Emergent Behavior”

• Incredibly seductive
• Witness ants

• Ants that hit obstacle turn right or left randomly!
Emergent Behavior

• Incredibly seductive
• Witness ants

• Ants prefer to follow a path with pheromones on it.

Emergent Behavior

• Incredibly seductive
• Witness ants

• Shortest path has more pheromones on it

Emergent Engineering

moveAround( X, Y, Point ) :-
    neighbor( X, Y ),
    brightness( X, N ),
    brightness( Y, M ),
    vacant( Y, Point ),
    N <= M.

• Attributes:
  - Ensemble level thinking
  - Concise understandable program
  - Scalable
  - Amenable to proof
  - Robust to failure and environmental uncertainty

For movie see: [www.cs.cmu.edu/~claytromics/movies/phototropic1.mpg](http://www.cs.cmu.edu/~claytromics/movies/phototropic1.mpg)
Thermodynamics for Computing

Aggregation fundamentally alters purpose and capabilities ⇒ causes a control barrier as systems scale

Traditional CS
- Control each unit’s actions

Engineering Ensemble Effects
- Control ensembles of active units
- Control global properties of the aggregate

Phy/Chem/Econ

Goal: Understand methods for programming the ensemble as a whole.

Solution Attributes

- Programs must be:
  - Ensemble centric
  - Uncertainty tolerant
  - Concise
  - Support formal methods

- Tools must:
  - Compile ensemble-program into unit program
  - Harness distributed nature of problem and solution

Some Recent Work

- Programming languages
  - Oriented to the ensemble
  - 20x shorter programs
  - At least as efficient!

- Algorithms
  - For shape morphing
  - Finding out where you are
  - Deciding how to get where you want to be

- Tools
  - For debugging distributed systems
  - Visualizing massive numbers of particles

For these and other simulations, see www.cs.cmu.edu/~claytronics/movies
**Something Concrete?**

- What programming model?
- What language?
- How to measure effectiveness?

- Imperative approach currently in use
  - Must program each unit
  - Must handle messages, links
  - Changing topology creates complications
  - Focused on the unit

**A simple “Walk”**

**C++** vs. **Meld**

```prolog
% C++
dist(S, D) :-
    at(S, P),
    D = |P - destination()|.

farther(S, T) :-
    neighbor(S, T),
    dist(S, D),
    dist(T, D'),
    D >= D'.
moveAround(S, T, U) :-
    farther(S, T),
    farther(S, U),
    T != U.

% Meld
dist(S, D) :-
at(S, P),
D = |P - destination()|.

farther(S, T) :-
    neighbor(S, T),
    dist(S, D),
    dist(T, D'),
    D >= D'.
moveAround(S, T, U) :-
    farther(S, T),
    farther(S, U),
    T != U.
```

**Example execution & Base Facts**

```prolog
neighbor(b, a)
neighbor(b, c)
at(b, point2)
neighbor(a, b)
neighbor(a, c)
at(a, point1)
```

```prolog
neighbor(b, a)
neighbor(b, c)
at(b, point2)
neighbor(a, b)
neighbor(a, c)
at(a, point1)
```

```prolog
neighbor(c, a)
neighbor(c, b)
at(c, point3)
neighbor(a, b)
neighbor(a, c)
at(a, point1)
```
Some Details

- Automatic distribution of facts, rules
  
  farther(S, T) :-
  neighbor(S, T),
  dist(S, D),
  dist(T, D'),
  D >= D'.

  farther-remote(S, T, D') :-
  neighbor(T, S),
  dist(T, D').

  farther(S, T) :-
  farther-remote(S, T, D'),
  dist(S, D),
  D >= D'.

- Handling deletion
- Messaging
- Side-effects

Can this possibly work?

- Amazingly:
  - Programs 20x smaller!
  - Message, bytes sent, memory used, and cpu used all scale at least as well as C++ program

Can this possibly work?

- Amazingly:
  - Programs 20x smaller!
  - Message, bytes sent, memory used, and cpu used all scale at least as well as C++ program

Global Behavior from local rules

- Concise specifications
- Embarrassingly parallel
- Examples:
  - Amorphous computing [Nagpal]
  - Graph grammars [Klavins]
  - Programming work [Kod.]
  - CA+Gradients [Stoy]
  - Hole motion [DeRosa]
  - Boyd model [Boyd]
  - Turing stripes

- Goal: Compile Global specification into unit rules
  Predict global behavior from set of unit rules

For movie see: www.cs.cmu.edu/~claytronics/movies/cubezman.mpg
My Vision for the future

- Imagine:
  - 3D Fax machines
  - Kinesthetic CAD tools
  - 3D TV
  - Surgical preparation
  - New drug design
  - Pario games
  - “In person” remote meetings
  - Doctors make house calls
  - Fire fighters out of harms way
  - ...

Imagine:
- Imagination is the main limitation
- Not just Claytronics!
- Give the power of programming to matter!

Challenge Problem(s)

- Intermediate uses for two classes of programmable matter:
  - 1mm particles, not very cohesive
  - 1cm particles, sort of cohesive
- Suggestions?