Beyond Audio and Video: Using Claytronics to Enable Pario

Seth C. Goldstein, Todd C. Mowry, Jason D. Campbell, Michael P. Ashley-Rollman, Michael De Rosa, Stanislav Funiak, James F. Hoburg, Mustafa E. Karagozler, Brian Kirby, Peter Lee, Padmanabhan Pillai, J. Robert Reid, Daniel D. Stancil, and Michael P. Weller

■ *In this article, we describe the hardware and software challenges involved in realizing claytronics, a form of programmable matter made out of very large numbers—potentially millions—of submillimeter-sized spherical robots. The goal of the Claytronics Project is to create ensembles of cooperating submillimeter robots, which work together to form dynamic three-dimensional physical objects. For example, claytronics might be used in telepresense to mimic, with high-fidelity and in three-dimensional solid form, the look, feel, and motion of the person at the other end of the telephone call. To achieve this long-range vision we are investigating hardware mechanisms for constructing submillimeter robots, which can be manufactured en masse using photolithography. We also propose the creation of a new media type, which we call pario. The idea behind pario is to render arbitrary moving, physical three-dimensional objects that you can see, touch, and even hold in your hands. In parallel with our hardware effort, we are developing novel distributed programming languages and algorithms to control the ensembles, LDP and Meld. Pario may fundamentally change how we communicate with others and interact with the world around us. Our research results to date suggest that there is a viable path to implementing both the hardware and software necessary for claytronics, which is a form of programmable matter that can be used to implement pario. While we have made significant progress, there is still much research ahead in order to turn this vision into reality.*

In this article we describe a concept for a new type of material, which we call *claytronics* (Goldstein, Campbell, and Mowry 2005), made out of very large numbers—potentially millions—of submillimeter-sized spherical robots. While still only a concept, we have completed a considerable amount of initial design and experimentation work, enough at this point to allow us to understand what is readily achievable within a short time frame (less than a decade) and also to identify some of the most significant technical challenges yet to be overcome. To date, we have developed and analyzed several promising engineering designs, conducted numerous large-scale experiments on a high-fidelity physics-based simulator, and successfully carried out several prototype three-dimensional (3D) microelectromechanical systems (MEMS) manufacturing runs. These experiences lead us to believe that there are no fundamental software or hardware barriers to realizing claytronics on a large scale and within a few years.

While the most fundamental purpose of our research on claytronics is to understand manufacturing and programming of very large ensembles of independently actuated computing devices, it is also clear that such a material would have numerous practical applications, ranging from shape-shifting radio antennas (important for software-defined radios) to 3D fax machines. Perhaps our most fanciful-sounding application, however, is motivated by one of the most basic of human needs: to communicate and interact with others. Two centuries ago, the only practical way to carry on a real-time conversation with

another person was to meet face-to-face, requiring both people to be in the same place at the same time. The invention of the telephone had a profound impact on society, and our ability to converse with people who are far away is something that we take for granted. Still, despite several generations of experience, using the telephone (or, today, the videoconference) is still not the same as meeting with someone face to face.

Hence we propose the creation of a new media type, which we call *pario*.[1] Similar to how audio and video allow us to render arbitrary sounds and moving images over long distances, the idea behind pario is to render arbitrary moving, physical 3D objects that you can see, touch, and even hold in your hands. As with audio and video, when we reproduce something with pario we are neither transporting the original object nor creating an exact replica: instead, the idea is to create a physical artifact that is a "good enough" reproduction of the shape, appearance, and motion of the original object—one that our senses will accept as being real.

As a concrete technology for pario, claytronics serves as a programmable form of modeling clay whose shape and appearance are remotely controllable, thereby creating physical objects that you can directly see and touch.

While we must admit that the notion of pario is fanciful, the practical impact of realizing it would be enormous. For example, consider how people collaborate to design 3D objects today (such as architects designing a house, or engineers designing a car). A mathematical representation of the 3D object is captured within the computer using some form of CAD software. While the ability to edit and share this representation is extremely powerful, how do the designers interact with this abstract model? Today, a monitor displays a (flat) perspective on the object from a particular viewpoint. To help imagine the object in three dimensions rather than two dimensions, a designer might spin the object around on the screen to see different perspectives. To modify the object, a designer reaches for a mouse or keyboard. The awkwardness of this interaction suggests that this is a case where people are bending to the limitations of technology, rather than technology rising to the level of what is most natural and useful to humans. While an improvement, virtual or augmented reality systems still limit user interaction, requiring the user to don special equipment, limiting the number of users that can interact with the environment. Most importantly, they do not allow the user to actually touch the artifacts.

In these two examples, the fundamental cause of the gap between what people want and what technology delivers is that computation is limited by its media types (for example, text, audio, video), and therefore is confined to "cyberspace." This either limits what technology can deliver or forces humans to adapt to technology in awkward ways.

Why Claytronics Is Well Suited to Meeting the Requirements of Pario

Taking pario from a fanciful vision to a practical reality will be extremely challenging. In this section, we begin by describing the requirements imposed by pario on any underlying technology that might be used to implement it, given the goal of creating a seamless pario experience. While there may be a number of different approaches to implementing pario, we describe in this section why we believe that the particular technology that we are pursuing—that is, claytronics—is an excellent match for the needs of pario.

Requirements for Pario

Pario requires an underlying technology that can construct a wide range[2] of high-fidelity dynamic physical objects at a large enough scale and with sufficient structural stability to enable hands-on human interaction. Note that existing 3D output technologies are insufficient for the following reasons. Although head-mounted displays or holograms can provide the illusion of a 3D object, no physical interaction with the object is possible.[3] While 3D printers (for example, fused deposition modeling)[4] have revolutionized the way that design and engineering takes place today, they can only render static objects, as opposed to the moving, dynamic objects necessary for pario.

We now consider each aspect of the pario requirements in greater detail to see how it translates to specific properties and requirements for claytronics. Later, in the claytronics hardware and programming sections, we describe our approach to realizing claytronics (including our current implementation status) in the context of meeting these requirements.

How Claytronics Meets These Requirements

We need to render a wide range of dynamic shapes, eliminating the possibility of implementing pario with a monolithic device. A monolithic device could not, for example, seamlessly create shapes that are not topologically equivalent. Hence the first requirement for claytronics is that it must be an ensemble of individual particles that work together to create the shape. We call each individual unit of the ensemble a *catom,* short for claytronic atom. Notice that the requirements of pario do not demand a particular base technology for these units: they could be cells constructed using synthetic biology or, in our case, catoms constructed using silicon and photolithography. To

support dynamic shapes (as well as shape formation), the catoms will need mechanisms for moving around each other. To maintain a given shape and to provide sufficient structural stability for physical human interaction, the catoms will need to be able to adhere to each other with sufficient force to maintain stability and exert force on other objects in the world.

We now consider the implications of wanting high-fidelity objects that can be constructed on a large enough scale for successful human interaction. Drawing an analogy to video technology, a successful video display must have sufficiently small, densely packed pixels to deliver a high-fidelity image, and the overall display size must be large enough for comfortable viewing. For pario, the individual catoms can be thought of as physical voxels (the physical 3D equivalent of pixels); we would like them to be small (preferably less than 1 millimeter in diameter) in order to deliver high 3D physical fidelity. To convey visual information, each catom needs to be able to change its color (similar to a pixel). If we also want to convey high-fidelity tactile information (for example, does an object feel like wood or glass when you run your finger across it), it should be possible to synthesize many textures by spacing the catoms on the surface in the proper arrangement, provided they are less than 0.3 millimeters in diameter (Klatzky and Lederman 2006, Lederman et al. 2006).

While high fidelity dictates submillimeter units, the requirement of constructing large enough objects for human interaction dictates that we will need massive numbers of them: on the order of millions. This constraint limits the kinds of manufacturing processes that can be used to build the units to one that is at least as scalable and cost-effective as a bulk-manufacturing method such as photolithography or synthetic biology. It also places constraints on the number of different kinds of elements that can make up the ensemble. We do not expect to achieve a seamless pario experience in the near term and instead aim at creating an ensemble of homogeneous submillimeter catoms that would be capable (in one form or another) of meeting the pario requirements.

The scale of the ensemble requires that computation be integrated into the ensemble. The main task of an ensemble at any given time is to reproduce a particular shape in three dimensions. Thus, at any given time each unit will have to execute a particular and, in general, different task depending on its current position. For example a unit might have to move to a new position, increase or decrease its adhesion forces along a particular vector, or change its color. The computation needed to compute these tasks is most easily located within the ensemble itself. This reduces the amount of communication bandwidth necessary to control the ensemble, and it increases the robustness of the ensemble. Contrast catoms to pixels on a display. In the latter case centralized computation works well. However, pixels only have state—they do not interact with each other. Catoms, on the other hand, need to interact with each other (such as when moving, sharing power, sensing the environment, and so on) in real time. If control was centralized, or even spread across a few locations, then communication between the catoms and the processor would introduce latency. Even for medium-sized ensembles, such a delay in the control loop would likely render the ensemble ineffective. Scaling to large ensembles would also become harder due to bandwidth constraints. Alternatively, consider the cells in our body: each one has some local "processing" and can perform significant functions independently. Of course, the individual units will have to be able to communicate with their neighbors and some must be able to communicate with external devices.

Another common feature of any technology that implements pario is that the individual units will require energy. Luckily, this does not mean that each unit must have its own independent source of energy. Instead, there must be two mechanisms to supply energy to a unit. At least some units must be capable of receiving power from an external source and transmitting that power to other units in the ensemble. The rest of the units must be able to receive (and transmit) power from (or to) their neighbors. To facilitate robustness and momentary disconnectedness, it would be advantageous if each unit could store some energy.

In the long run it will most likely be advantageous for the ensemble to be made up of a small set of specialized units. For example, some might be specialized for computation, others for external communication or energy storage. The units may even have different shapes to support, for example, the building of complex mechanisms. In the Claytronics Project we have chosen to limit our investigations in the near term to ensembles made up of homogeneous units, thus significantly simplifying the programming problem as all the units are interchangeable.

Claytronics Hardware

At first glance the ability to create a coherent ensemble of millions of units that meet the above requirements appears fantastical—almost science fiction. But, if we step back and examine it, the question is not *if* we can manufacture it, but *when*. It is clearly possible to do so in principle; for example, biology builds ensembles of units that coordinate together to form dynamic 3D shapes that can interact in the real world, and we already have an entire industry based on bulk manufacturing

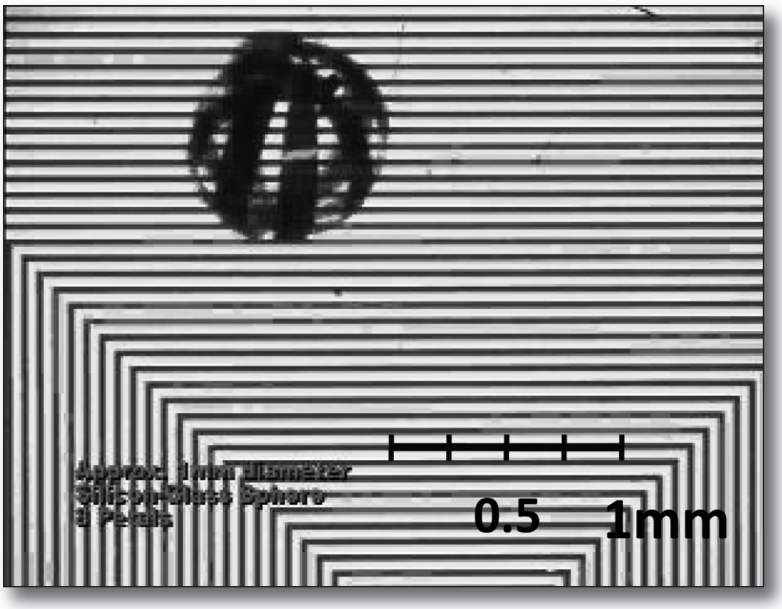

Figure 1. Spherical Shell.

An example of a spherical shell (with diameter of approximate 0.9 millimeters) made using standard photolithography and stress-induced curling. The shell (in the upper left) is sitting on a circuit board that can move the shell using electrostatic forces.

today—the semiconductor industry has been building computer chips for decades using photolithography. The same basic process used to manufacture chips has also been used to commercially create MEMS circuits that are integrated on the same die with mechanical systems (Madou 2002). MEMS processes can also be used to create 3D devices. Reid has constructed spherical shapes by first printing a projection of the sphere and then, by harnessing the inherent stresses in thin film silicon dioxide, causing the projection to fold up into a sphere (see figure 1) (Reid, Vasilyev, and Webster 2008; Vasilyev, Reid, and Webster 2008). This same process can be applied to a prefabricated CMOS wafer to create 3D units with integrated processors and actuators.

As we hinted earlier, MEMS represent just one possible implementation technology. As the individual units scale down in size, other technologies that can produce less expensive units will have to be used. For example, researchers at Intel have shown that the submillimeter spherical shells can be formed using inexpensive materials in silicon molds.[5] In the long term, approaches based on synthetic biology (Endy 2005) are very appealing.

Ensemble Principle

Independent of the implementation technology there are some common requirements imposed by our desire to create ensembles of millions of cooperating units. Because our goals are oriented around the ensemble as a whole, units only need to function when they are part of the ensemble. This leads to one of our guiding engineering principles, the ensemble principle: an individual unit should include only enough functionality to contribute to the desired functionality of the ensemble. Keeping this principle in mind forces us to simplify the hardware and software, ideally to the point where each unit is as simple as possible, enabling inexpensive robust units and thus more robust ensembles. A concrete example of this relates to how the units move. Note that the requirements listed in the previous section do not require that the individual units move independently. In fact, there is nothing that says that the individual units even need to include a mechanism for locomotion. If an external force, for example, air currents, is applied to the ensemble, it is possible in principle for the units to reform into a new shape by controlling when and where they adhere to each other. By letting go at one place and then sticking at another they can form arbitrary shapes. A less extreme example would incorporate a mechanism that can be used to allow cooperating units to move, but not allow an individual unit to move without cooperating with other units in the ensemble. For example, in our early prototypes the individual units used electromagnets for movement (Kirby et al. 2007). Units move when two neighboring units each energize their magnet in the appropriate direction. This also allowed us to simplify the individual units by eliminating all moving parts.

Basic Catom Mechanisms

In the remainder of this section we describe one possible path to realizing claytronics. The mechanisms discussed here are not intended to be final solutions, but rather to show that at least an early version of pario can be realized using techniques and methods that are understood today. After presenting our current approach we briefly discuss some of the more challenging aspects to building a catom.

Using the ensemble principle as a guide we show how a monolithically manufactured unit can compute, communicate, move, adhere, sense, and share power with other units using just two basic mechanisms: a processor and an arrangement of conductive plates under the surface controlled by the processor. As will soon be clear, the smaller the catom, the easier it is to actuate. However, the catom needs to be large enough for the processor, storage capacitor, and other circuits. For this discussion we will assume that the diameter of the catom is 0.7 millimeters. This gives us a total usable area for circuits of just over 1.5 square millimeters.

For the reasons given earlier, each unit must contain a processor and memory. Even using older 90 nanometer technology we could include a core similar to the ARM7,[6] 64 kilobytes of nonvolatile memory and 64 kilobytes of double data rate dynamic random access memory (DRAM) in 0.3 square millimeters. A clock rate of 30 kilohertz for approximately 0.03 million of instructions per second (MIPS) would be sufficient to run the programs necessary for ensemble behavior and allow us to keep nominal power consumption at 1.8 microwatts. Of course, as process technology improves we could reduce the size of the catom or increase processing power or memory capacity and potentially lower the energy requirements. The numbers and examples we cite here and in the remainder of the section are used only to show that even with current technology it is possible to meet the needs of a catom.

We use electrostatic forces for catom movement, adhesion, communication, sensing, and power distribution. The electrostatic forces are generated by having the processor route charge to conducting plates that are printed on the die. These plates will be just under the surface of the catom beneath a dielectric layer of silicon dioxide (SiO_2) (figure 2). The geometry of the catom ensures that when two catoms are adjacent to each other they will create capacitors between adjacent electrodes. Since catoms cannot share a common ground, catoms are coupled by using pairs of electrodes. When a voltage is applied between the plates of a capacitor, attractive forces are created due to the accumulated charge on the plates (see figure 2). This same basic mechanism is used for all the electrostatic-based functions. Note that our use of electrostatics for adhesion, movement, sensing, and power distribution is an example of the ensemble principle. The mechanism is very simple, yet no single catom can move, and so on, without interaction with other catoms.

Motion of the whole ensemble occurs when large numbers of individual catoms move themselves around their neighbors. Individual catoms only move by rolling around their neighbors into a vacant space, either on the outer surface of the ensemble, or into a void in the interior of the ensemble. This limits the number of catoms that can move at any given time placing constraints on the ensemble motion-control algorithms. However, algorithms such as those described in De Rosa et al. (2006) and Dewey et al. (2008) operate with just this sort of behavior. This form of motion requires that actuation be sufficient to move only one catom. Since the moving catom does not rub against any others, friction is negligible. When the catom on top is to roll around another catom clockwise (see figure 3 for a profile of the movement), a voltage difference is applied between all

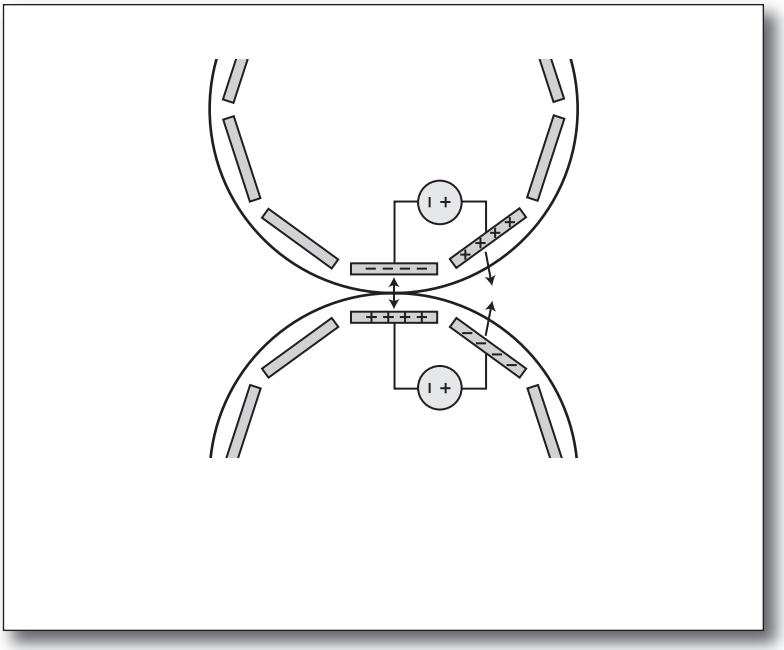

Figure 2. Electrodes Are Placed Radially on the Catom Surface.

the plates that are located in the bottom right quadrant of the upper catom and the electrodes that are located at the upper right quadrant of the lower catom, as shown in figure 2.

The force generated by the electrostatic plates is a function of catom diameter, electrode size and spacing, and applied voltage. In designing a catom of a specified size (in this case 0.7 millimeters diameter), the force generated at a fixed voltage is first calculated as a function of the number of electrodes resulting in the plot like that shown in figure 4. Based on the plot, the optimal number of electrodes for a 0.7 millimeter diameter sphere is 43, or roughly one electrode every 8–9 degrees. Using this electrode count, the voltage required to move the catom vertically against gravity (assuming the catom has 1/13 the density of water) is approximately 94 volts. This voltage decreases with the catom diameter because as the catom scales down in size, the torque required to move against gravity decreases faster than the torque generated by the electrostatic force. For a catom with a 0.5 millimeter diameter the required voltage is 60 volts. This then suggests that the smaller the catom, the better.

In the near term, practical considerations push for larger catom sizes. A catom diameter of 0.7 millimeters provides a nice compromise. The easiest way to move the catoms would be to have them move one diameter and then come to a complete rest before moving again. With the assumptions made above and assuming that a catom comes to complete rest after moving one diameter, catoms should be able to move at least 20 body lengths a

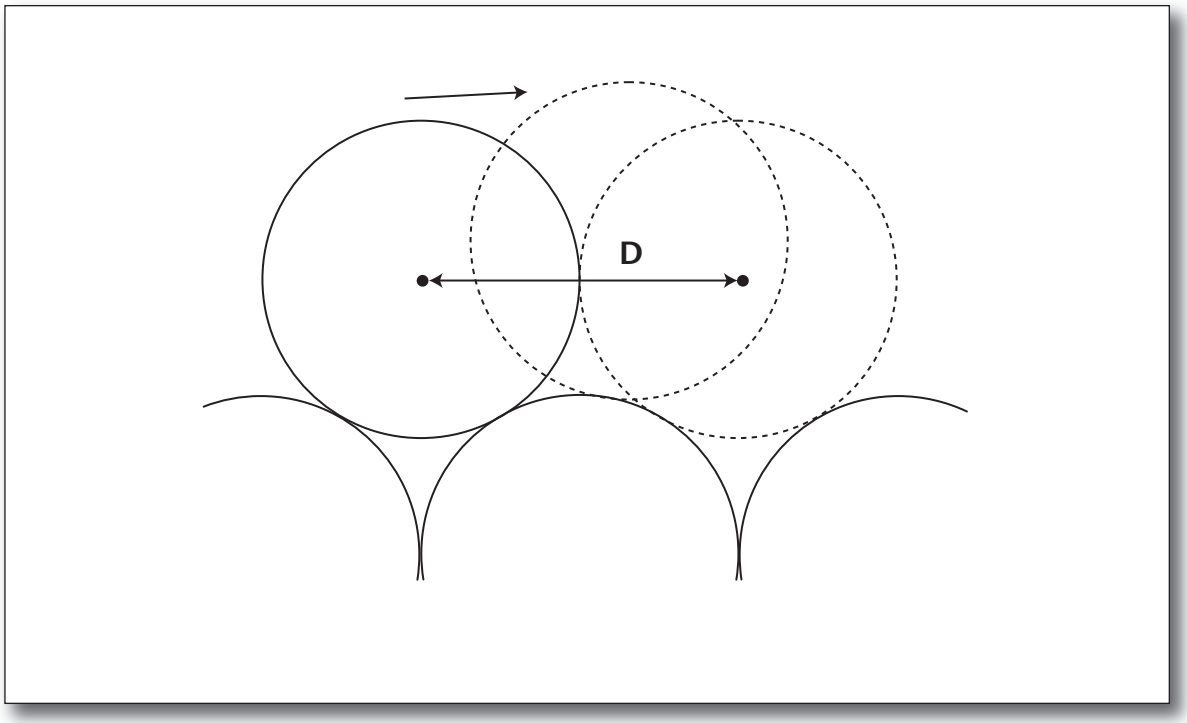

Figure 3: The Motion Profile of a Catom Moving along the Ensemble.

second horizontally and at least 10 body lengths a second against gravity. This is far short of the speeds needed for generalized pario, but reasonable for nearer-term applications. For faster operation, dynamics and drag need to be taken into account.

Electrostatics can also be used to cause adjacent catoms to adhere to each other. In this case, the electrodes on each catom are oppositely charged. Again, using the above assumptions, electrostatic-based adhesion should allow a catom to carry the weight of at least 10,000 catoms hanging from it, or a chain of catoms 8 meters long. Of course, such a structure would be incredibly frail. In practice large structures will have many layers that combine to resist external forces and the like.

Power can be shared between catoms using the same mechanism. Much as a transformer uses an electromagnetic coupling between coils to transmit power across an insulator, a pair of capacitively coupled electrodes can be used to carry energy across a nonconducting gap. Figure 5 shows the basic idea—an alternating current (AC) signal is put on the plates on the catom with power, and the capacitive coupling causes mirror charges to appear on the adjacent catom. This signal is rectified and stored in a local storage capacitor. Analysis similar to that described by Karagozler et al. (2007) shows that using this mechanism, power can be transferred with between 50 percent and 66 percent efficiency in less than 40 milliseconds for a millimeter-scale catom. The lack of efficiency of this mechanism requires us to explore alternative means of providing power to the vast majority of catoms. Among the mechanisms we are exploring is providing power through transmitted energy such as radio frequency (RF) or optical. In the RF space we are looking at magnetic resonant coupling (Cannon et al. 2008). In the optical space we are examining small solar cells that can fit into the catom. Such a cell with cross-sectional area of 0.25 square millimeters can provide 15.7 microwatts (solar mass is equal to 1.71, 8.3 percent efficiency). Conservatively, our baseline design is done with a 10 microwatt power budget.

A simple low-power communication mechanism would modulate the voltage on the actuator plate adjacent to the neighboring catom. The modulation is accomplished indirectly by charging and discharging a smaller plate embedded below the main actuation plate. This isolates the smaller plate from high voltage on the main plates. It also has a negligible effect on the adhesion forces needed to maintain ensemble cohesion. Driving the communication plates will capacitively propagate the signal to the neighbor's plates. The energy required to communicate in this way is, in the worst case where we dump the charge after every bit, about 0.7 picojoules per bit. Thus, even a 1 megabit channel would require less than 1 microwatt. This method uses very little power but is susceptible to noise that may arise from mechanical vibration of the catoms. A more power-inten-

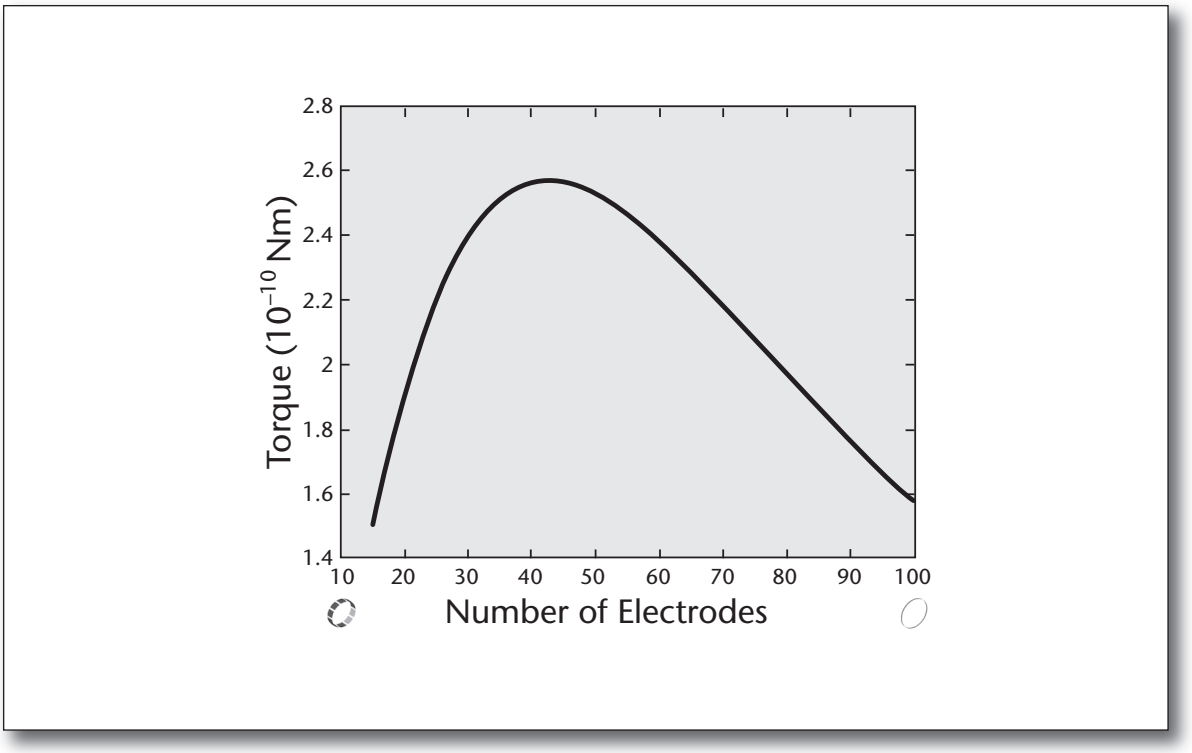

Figure 4. Generated Torque Versus Number of Uniformly Distributed Electrodes for a 0.7 Millimeter Diameter Catom at 100 Volt Bias.

sive communication method would modulate a carrier that would add an additional 1.7 microwatts.

The communication plates can also be used to sense the presence of neighbors. Additionally, if the surface of the catom has any conformance, increased pressure between adjacent catoms will be detectable as an increase in capacitance. If this proves insufficient we can embed piezoelectric strain sensors in the silicon shell of each catom. It should be noted that in either case, no single catom will be able to accurately determine the direction and amount of force being applied. Instead, they will have to communicate with each other to effectively create a synthetic sensor.

The final major piece necessary for even an early version of pario is for each voxel to be able to change its color. Cognizant of the power requirements of a display, and to give claytronics a more real appearance, we have elected to pursue a reflective approach. Each catom will be multicolored. The catom will rotate in place to present the proper color to the user. This approach means that no power is required to maintain a particular color.

The plan outlined here requires a nominal power budget of less than 10 microwatts. Commercially available EIA size 0201 capacitors[7] would provide enough energy to run for about 250 milliseconds. At this power level the unit can maintain a reasonable operating temperature through black body radiation. If we use capacitive coupling for power sharing, this means that in the worst case the catom spends more than 80 percent of its time doing useful work.

Significant Challenges

The design sketch presented above shows that all the functionality required to implement claytronics, at least for medium-scale ensembles, is possible. However, there remain many engineering challenges before catoms are a reality. Table 1 gives a brief outline of the path we are taking toward implementing claytronics along with the status of the individual components.

As expected, the processing and communication hardware is sufficient for even the most challenging applications. Furthermore, as semiconductor fabrication continues to improve we will have even more resources capable of satisfying the computing needs of claytronics. Supercapacitor technology continues to improve, but even today's commercially available packages appear to provide sufficient storage capacity. The display mechanism also appears to be sufficient, given small enough catoms. Finally, the electrostatic-based actuation mechanism, combined with the right control software, will give the catoms sufficient capability and speed for movement.

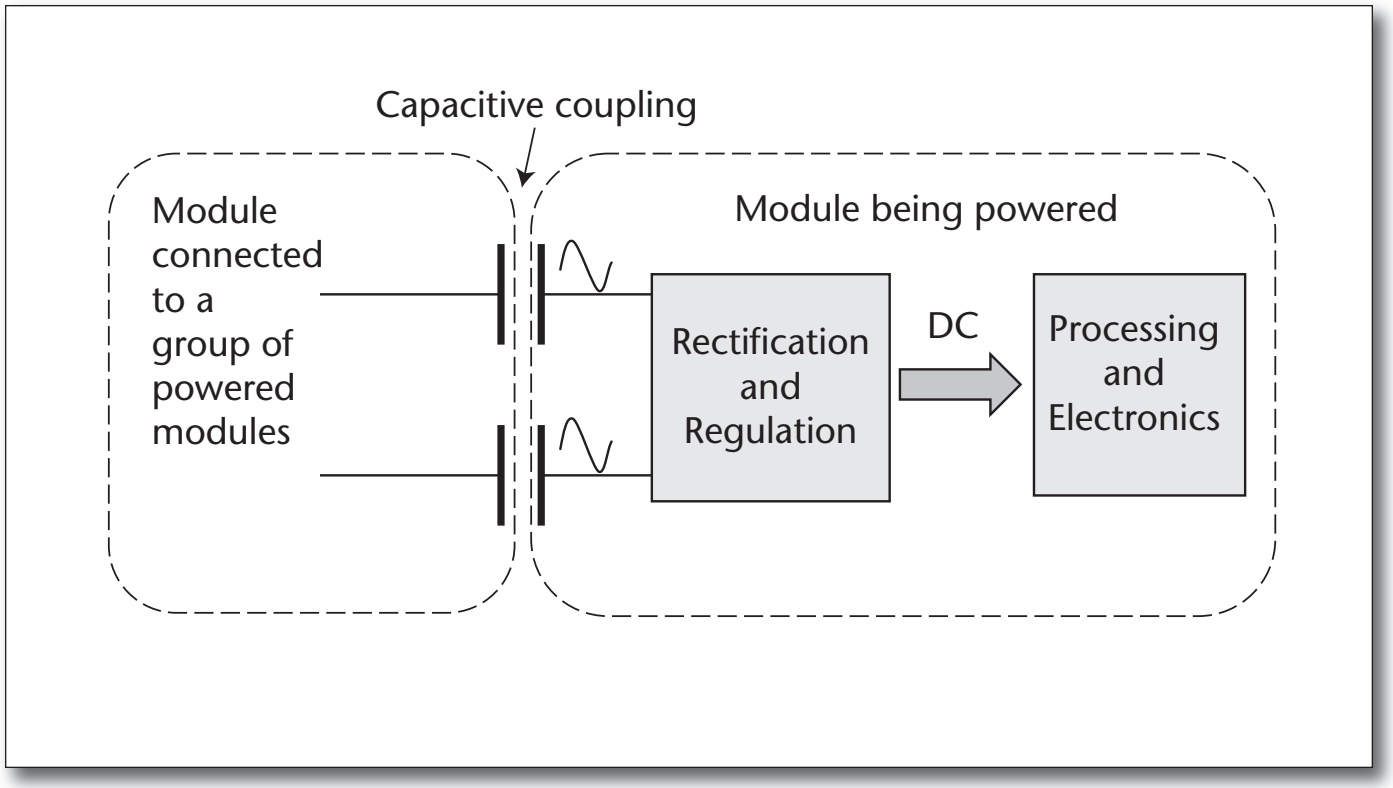

Figure 5. Scheme for Transferring Power between Catoms.

From a hardware point of view, the two key challenges are power distribution and adhesion. Capacitive coupling, which is a good starting point for both, is not sufficient to scale to large ensembles. In the case of power distribution, capacitive coupling is not efficient enough to transmit power through a long chain of catoms. We are currently investigating several alternatives including photovoltaics and magnetic resonant coupling (Kurs et al. 2007, Cannon et al. 2008). In both of these cases, the catoms can be essentially made transparent. Thus, only the most central units—the ones with the least to do—will require capacitive coupling for power. For adhesion, a straightforward application of capacitive coupling is impressive and good enough for many near-term pario applications, but it will not be sufficient to create the kinds of arbitrary structures necessary for generalized pario. Solving this problem will likely require drawing on reversible chemistries, that is, chemical process that involve the reversible formation of chemical bonds without side reactions.

Claytronics, as we have described it here, is made up of a homogeneous ensemble of units. We believe that this is the proper place to start exploring programmable matter, but not the ending place. If rendered objects are going to move in a natural way, catoms cannot just flow around each other. Instead, they will need to assemble into components that have some relationship to the object they are rendering. In the case of a human, this would involve creating joints, bones, and muscles. In all likelihood this will require several types of units in the ensemble.

Programming Claytronics

There are significant challenges in programming any distributed system. Programmable matter poses additional challenges that arise from limited resources, an ever-changing topology of network connections that result from the units moving around, and high action-uncertainty as a result of the constant interaction with the physical world. The programming challenges can broadly be divided into two areas: programming the individual units and programming the ensemble.

The issues involved in programming an individual unit are essentially the same as those related to programming individual robots, particularly modular robots (Yim et al. 2007) (that can be viewed as an early form of programmable matter). The critical difference between modular robotics and the claytronics vision is that of scale: we must consider many more much smaller robotic modules. Very small units pose challenges due to lesser capabilities, in terms of programming resources, noisier sensing, and lesser physical actuation capability.

Capability	Status	Description
Individual Actuated Unit		
Physical Structure	Demo-mm	Basic stress-induced sphere creation demonstrated (Reid et al. 2008). (Other methods also demonstrated[5] or in the design phase.)
Energy Collection	Demo-cm	Demonstration of capacitive-based mechanisms at macroscale (Karagozler et al. 2007, Karagozler 2007); Design and analysis of photonic based collection; Demonstration of cm-scale resonant coupling mechanism (Cannon et al. 2008)
Energy Storage	Commercial	Capacitors of the right volume and capacity are commercially available.
Actuation	Demo-mm	Demonstrated at the mm-scale (Karagozler et al. 2007[9]).
Adhesion	Demo	Electrostatic-based adhesion demonstrated at the macroscale and mm-scale (Karagozler et al. 2007[9]); Fiber-based adhesion demonstrated at centimeters and below (Murphy et al. 2007a, 2007b).
Integration	In Process	Tests integrating the above are being carried out.
Programmable Actuated Unit		
Processing	Commercial	Commercially available cores will fit in area and energy budget with sufficient processing requirements.
Memory	Commercial	As above.
External communication (input only)	Design	The capability to recieve data from the outside for programming and control of the units.
Small Ensembles		
Contact sensing	Design	Using changes in capacitance on surface.
Neighbor-to-neighbor communication	Design	Using capacative coupling as described above.
Large Ensembles		
Energy Transfer	Demo-cm	Transfer of power using capacitive coupling demonstrated at the cm-scale (Karagozler et al. 2007, Karagozler 2007).
Generalized Pario-capable Ensembles		
Color	Design	As described above.
Specialization	Ideas	Specialized units for power storage, external communication, and possible mechanical structural.

Table 1: Current Status of the Mechanisms Necessary and the Design Path for Claytronics.

Status key: Commercial: Already available commercially; Demo-mm: Demonstrated at the mm scale; Demo=cm: Demonstrated at the cm scale; Design: Paper designs, simulations, and analysis—yet to be demonstrated; In Process: Currently working on demonstrating; Idea: Still working on the design.

Our main focus has been on the problem of programming the ensemble as a whole. One of the most challenging aspects of programming ensembles with several orders of magnitude more units than typically considered in modular robotics is to find and express scalable algorithms. One of the main impediments to this, as in any massively distributed systems, is that the programmer must write code for each node in the system individually, and yet ensure the overall distributed system reaches the correct high-level result.

Traditional imperative programming languages, such as C, C++, and Java, do little to address ensemble-level programming issues. These lan-

guages are inherently oriented towards a single processing node and require significant additional effort when used in a distributed setting. In addition to creating a representation of the data needed for an algorithm, the programmer must determine what information is available locally and what must be obtained from remote nodes, the messages and protocol used to transfer this data, mechanisms to route or propagate information through multiple hops as needed, and a means to ensure the consistency of this data. Furthermore, in algorithms to control ensembles, it is almost always necessary to express the conditions and actions that need to be carried out in terms of information that spans multiple units. Languages that constrain the programmer to the perspective of a single node make such algorithms difficult to implement.

Successful approaches to this programming problem must in the end deal with but one issue, scalability. The first condition required to ensure scalability is to allow the programmer to think of the ensemble as an ensemble, and not as many individual nodes. This allows the programmer to focus on the overall problem, not the details of the individual nodes. We push onto the compiler the task of compiling the ensemble-level description into low-level code for each node. Next, the language should support a concise, but understandable, expression of the algorithm. We use program length as a metric because shorter programs are generally easier to understand, modify, debug, and optimize. Scalability also requires uncertainty tolerance to handle the failures inherent in a system with large numbers of nodes working in the physical world. Finally, to cope with the exponential number of states and transitions in such a system, the language and tools should support formal methods for proving the program is correct.

One path to programming ensembles is to explore emergent approaches such as cellular automata. This is seductive because often a small set of simple rules can result in interesting and complex behavior. However, despite significant effort in this area, there is still no automatic way of determining, in general, the behavior that will arise in an ensemble from the local rules of the agents. Instead, we take the opposite approach, to specify high-level global behavior that is compiled down into local programs to be run on the units of the ensemble. While such an engineered approach may not derive the smallest set of rules necessary, there are hints (based on recent research) that it is possible. This is particularly true in the case where people have investigated programming methods to render shapes in massively distributed systems of interacting agents (Jones and Matariae 2004; Zykov and Lipson 2006; Klavins, Ghrist, and Lipsky 2006; De Rosa et al. 2006; Abelson et al. 2000; Nagpal 2001; and Stoy and Nagpal 2004). Thus, we believe that pario—whose main goal is to render 3D objects—is a good place to start to understand the problem of programming programmable matter.

Ensemble Algorithms

All of the activity in claytronics revolves around the ensemble as a whole. In keeping with the ensemble axiom, the individual units are very simple and not fully capable of functioning on their own. Rather, a group of units or even the entire ensemble needs to work in concert for many essential functions. Even to bootstrap and power up a claytronics ensemble requires collaboration. An ensemble placed on a powered surface (such as a grid of power and ground lines) will have access to power only on its surface. We have developed a partially directed, semirandomized algorithm (Campbell, Pillai, and Goldstein 2005) that allows a few powered units to generate and extend a power grid throughout the ensemble, eventually distributing power to all the units. We have leveraged emergent behavior in our approach, since the units start without any power and are initially able to run only very simple programs.

Once the nodes are all powered, they can run more complex code. The functions and actions that must be performed by each unit for pario are often determined by the position of the unit in the ensemble. Hence, the next task is for the units to establish an ensemble-wide coordinate system. They do this by sensing where their neighbors are relative to their own coordinate frame and by running a distributed algorithm that determines a consistent coordinate system across the ensemble, even when the sensor readings are very noisy. Our algorithm for this internal localization task performs global pose estimation of the units by hierarchically partitioning the ensemble into smaller components, recursively applying the algorithm to the smaller pieces, applying rigid alignment to merge partial solutions, and using distributed, iterative refinement to smooth the results. This algorithm has been shown to determine the locations of the catoms in large ensembles to within 3 percent of their true position, when each module has 12 sensors around its perimeter (Funiak et al. 2008). Furthermore, it is highly scalable, that is, the communication complexity of the algorithm scales logarithmically in the number of units. In addition to providing power and performing internal localization, the ensemble may at this point undertake other startup procedures, such as establishing a network overlay for efficient communication, or constructing a picture of the environment by combining the sensed information from multiple surface units.

At this point the ensemble is ready to morph

```
// Calculate the distance to the target
dist(Catom,Distance):- at(Catom,CatomLocation),
    TargetLocation = destination(), // Retreive the destination
    Distance = |CatomLocation - TargetLocation|, // Determine distance to destination for Catom
    Distance > robot-radius.   // Stopping condition, that is, Catom at destination

// Determine which module is the farthest away
farther(FarCatom,NearCatom):- neighbor(FarCatom,NearCatom),
    dist(FarCatom,Distance_FarCatom),
    dist(NearCatom,Distance_NearCatom),
    Distance_FarCatom ≥ Distance_NearCatom.

// Move MovingCatom around PivotCatom until it touches TargetCatom
moveAround(MovingCatom,PivotCatom,TargetCatom):- farther(MovingCatom,PivotCatom),
    farther(MovingCatom,TargetCatom),
    TargetCatom ≠ PivotCatom.
```

Figure 6. Walk Program in Meld.

into a particular shape. Our current approach to ensemble shape change uses a generalized metamodule planner (Dewey et al. 2008) that operates on subcollections of modules (also called metamodules) that have fewer motion constraints than the individual modules. This allows for efficient, distributed shape planning, with a lower-level controller converting metamodule motion primitives into a sequence of module motions. In our system, a 3D model of the desired shape is broadcast to the ensemble and then distributed among the catoms. Each of the catoms then executes a small distributed program that attempts to create or destroy metamodules to achieve the goal shape. In coordination with local neighbors, the catoms move to effect the creation and deletion of metamodules, thus morphing the ensemble to the target shape. Although our algorithm is nondeterministic, its implementation has been proven to be both sound and complete: barring failures, the program will not disconnect the ensemble, and if the target shape can be reached, then it will eventually be reached.

These three ensemble tasks illustrate some of the different approaches one can take in creating programs for an ensemble. The power distribution and shape change use emergent techniques, and rely on a few simple rules with local sensing and communication to achieve a fairly complex task. In general, it is very difficult to determine a set of local rules to actually cause a desired ensemble-level behavior, but once such rules are constructed, they are easily implemented and executed on the individual nodes. The localization program, on the other hand, uses a more top-down approach, with an algorithm developed independent of the local operations of the modules and then translated to run on the ensemble. Implementation of this algorithm is much more involved, requiring significant cross-ensemble coordination, communication, and synchronization to work effectively. On the other hand, this approach allowed us to effect the desired complex, multistage behavior across the ensemble.

Programming Languages for Claytronics

In pursuit of our goal we are exploring different programming approaches and have developed two new programming languages: LDP (De Rosa et al. 2008, Ashley-Rollman et al. 2007a) and Meld (Ashley-Rollman et al. 2007b). Both of these languages are declarative in nature and result in programs that are about 20 times shorter than equivalent imperative programs. They each take an ensemble perspective, allowing a programmer to create simple, concise programs that are automatically compiled down to programs that run on each unit. In the examples shown in figures 6 and 7, each language is used to implement a simple three-node locomotion algorithm, whose steps are illustrated in figure 8.

Meld is a declarative logic programming language based on Datalog (Ceri, Gottlob, and Tanca 1989) and inspired by P2 (Loo et al. 2006). A Meld program consists of facts and the rules for deriving them. A fact represents the current program state

```
scalar radius;
point location,destLocation;
scalar distance = INT_MAX;

#update the distance from a to destination, if a is not already there
forall(a) where (a.distance > a.radius) do
            a.distance = |a.location - destLocation|;
#if a is the farthest catom, it rotates around c until it contacts b
forall(a, b, c) where (a.distance > b.distance) & (a.distance ≥ c.distance) do
           a.moveAround(c.id,b.id);
```

Figure 7. Walk Program in LDP.

including observations about the world, actions that should be performed, the goal or result of an algorithm, and any internal algorithm state. The facts representing observations and those that perform an action are used to interface the ensemble with the environment. The observation facts include sensor data as well as information about the network topology of the catoms. The network topology information (neighbor facts in the example in figure 6) is especially important to both Meld and the programmer. Meld uses these facts to automatically distribute data throughout the ensemble. The programmer needs these facts in order to effectively understand the physical geometry of the ensemble. The action facts activate an actuator when derived; for example, moveAround causes the catom to rotate around a neighbor.

A Meld program, such as the one shown in figure 6, is executed through a process called *bottom-up reasoning* or *forward-chaining*. The observations about the world (neighbor and at) constitute the starting set of known facts. The rules of the program are matched against these facts to derive new facts that are then added to the set of known facts. When an observed fact changes it is removed from the set of known facts through a process called *deletion*. Deletion continues until all the derived facts based on the old deleted fact are also deleted. This allows the program state always to reflect the current state of the world, simplifying coding for the programmer and providing an automatic means for discovery of and recovery from failures. The programmer only needs to specify the high-level logical relations and reasoning rules; the compiler takes care of the low-level operations: efficient communication of facts between the units, application of the rules, ensuring forward progress on proofs, and deletion of derivations when facts are refuted or deleted. The programmer is freed from the burden of determining what messages to send and to what units, and from managing of data at each node. Algorithms that require very long, complex programs in C or C++ style languages can be expressed in just a few lines in Meld.

Locally distributed predicates (LDP) is a declarative programming language derived from distributed watchpoints (De Rosa et al. 2007), a debugging facility designed to identify and detect multinode error conditions in ensembles. LDP allows programmers to specify distributed conditions among small, connected groups of nodes, and to trigger actions when groups matching the condition are detected. LDP treats each node as a collection of named state variables and allows for the expression of predicates that combine node state, historical state, and topological constraints. The underlying LDP run time uses radius-limited distributed snapshots to continuously search for matching groups of nodes. Once a predicate matches it can trigger arbitrary actions. These actions can include modifying state variables, calling arbitrary C functions, and rearranging the node's local topology through movement. Though LDP provides fewer proof properties than Meld, the LDP run time is designed to integrate easily with lower-level C code, providing (for instance) access to sensor readings directly as state variables. With the exception of such low-level interactions, all variable storage and messaging is handled by the LDP run time, allowing for extremely concise expression of distributed algorithms.

Current State of the Art

Both Meld and LDP have been shown (Ashley-Rollman et al. 2007b, De Rosa et al. 2008, Ashley-Rollman et al. 2007a) to create programs that are significantly smaller (approximately 20 times) than their imperative counterparts. For example, a com-

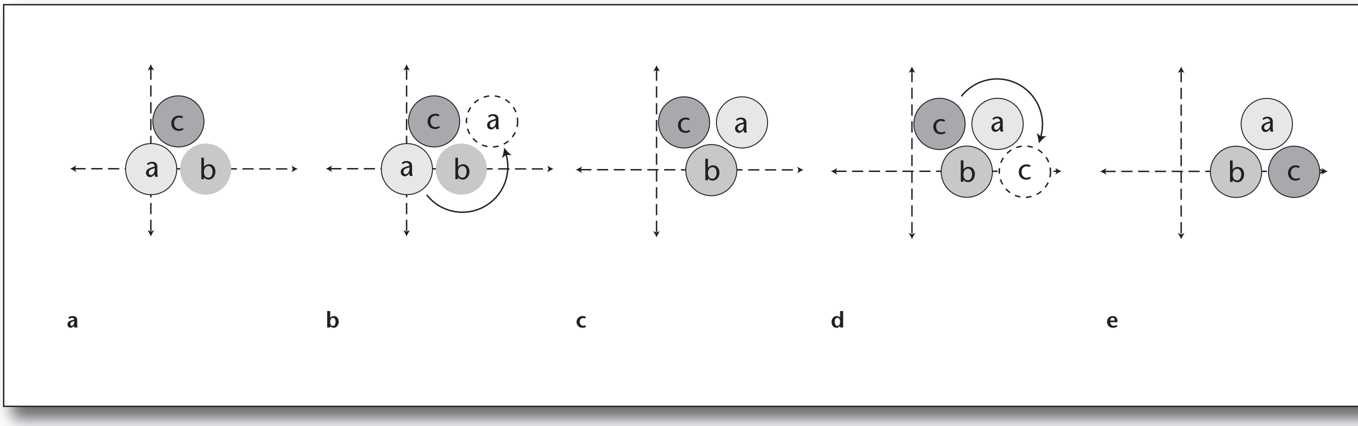

Figure 8. Illustration of Catoms Collectively "Walking."

a. Starting location at origin; b. Catom a moves around b; c. Catom a finishes moving; d. Catom c now moves around a; e. Catom c finishes moving.

plete shape planner, as described in Dewey et al. (2008) is implemented in just 28 lines of LDP code.

One of the advantages of concise programs is that it is possible for the programmer to think about the whole program, which facilitates correctness and optimizations. We see this in the performance of the implementations. Comparing Meld and C++ to implement applications has shown that the Meld compiler is effective in distributing the application among the nodes (Ashley-Rollman et al. 2007b). In the case of larger programs we often see better performance in the Meld implementation. The main reason for this is that the Meld implementation makes use of much of the latent parallelism inherent in the algorithm, while the C++ implementation is limited to the parallelism that the programmer can manage and explicitly encodes.

Our investigations into fault tolerance and automatic verification of programs is still in its infancy. Preliminary results indicate that the ensemble-centered programming styles embodied in LDP and Meld lead directly to fault-tolerant programs. In Meld, for example, when a unit fails to move to the correct location, the system automatically removes any derivations that do not agree with the current location of the unit. We have shown that running our shape planning algorithm on one million units still results in success even when more than 10 percent of the units fail during execution.

In the area of automatic verification, we have been able to prove important properties of our programs using a combination of paper proofs and a straightforward manual translation from Meld source to inputs for theorem-proving tools like Twelf (Pfenning and Schurmann 1999). This permits us to prove correctness properties about algorithms and their actual implementations. For example, we have proven that our general shape-planning algorithm (Dewey et al. 2008) written in Meld is complete (that is, if the target shape is reachable, it will be created) and sound (that is, the ensemble will never become disconnected).

Research Challenges

Significant research challenges remain in the methods and languages used to program programmable matter. Both of the declarative languages we have developed treat the ensemble as a set of networked processing nodes. This has worked well for the relatively low-level tasks we have thus far investigated, but it is not clear if this is the right abstraction for higher-level tasks. For many pario applications, it may be desirable to have direct language support (for example, for spatial information and temporal dynamics) to describe the high-level shapes and motions desired. This may require abstractions similar to those used in 3D graphics programming or modeling. Whether a completely new language is needed for this or if these abstractions can be built on top of our existing declarative languages (such as with the metamodule abstraction used for shape change) remains an open question.

Pario Application Scenarios

Having described some of the technology behind claytronics that might be used to implement pario, we now present an application scenario to illustrate how pario might be used in practice at some point in the future. The following scenario will appear, at first glance, to be pure fantasy; however, as we described earlier, there is a path toward realizing a technology (claytronics) that will enable pario.

When Dr. Alice Smith saw the extent of the chest

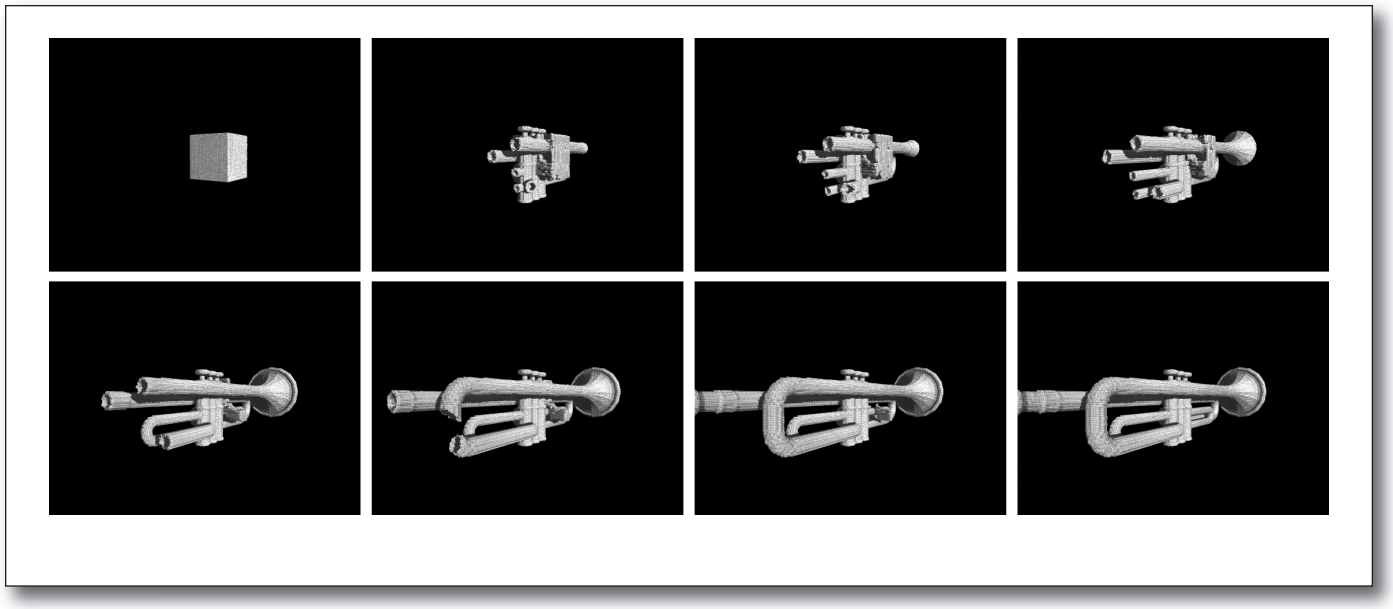

Figure 9. A Trumpet Being Formed.

Some snapshots of a trumpet being formed from an ensemble of one million catoms running a generalized metamodule planner (Dewey et al. 2008) on our simulator.[8]

injury to Fred, who had just been rushed into the emergency room, she knew that she wanted to have an immediate face-to-face meeting with two of her colleagues: Bob and Charlie. At this moment, Bob is at home, and Charlie is working in a hospital in a different city. Fortunately, Bob's house and both hospitals have rooms that are equipped for "pario-conferencing," which means that they contain relatively little furniture, a sandboxlike container full of claytronics, and an array of cameras discreetly placed in the walls and ceiling for performing 3D capture. Alice initiates the meeting by sending electronic invitations to Bob and Charlie through the Internet (similar to how one initiates a video chat session today). Once Bob and Charlie accept these invitations, a digital connection is established between the three rooms that will be used to keep them synchronized.

The camera array in Alice's room captures the shape and appearance of Alice and any furniture or other objects in her room, and a corresponding process occurs in Bob's and Charlie's rooms. After software confirms that it is safe to do so (for example, there won't be any collisions), the claytronics material in Bob's room forms into the shapes of Alice and the chair that she is sitting on, and colors itself to take on their respective appearances. The same thing happens in the reverse direction: a copy of Bob and his furniture are rendered by means of claytronics in Alice's room. Finally, both Alice and Bob are rendered in Charlie's room, and Charlie is rendered in both of their rooms. The net effect is that each room will contain the same sets of physical objects, where some objects are the originals and others are pario replicas from other rooms. As people or objects in any of the rooms move, their replicas also move accordingly as they are continuously updated based upon the input from the camera arrays.

The performance requirements of this "face-to-face" meeting scenario are perhaps the most challenging of any pario application (given the desire for realistic motion and real-time responsiveness of humanoid objects); hence we expect this aspect of the application scenario to have the longest time horizon until it is practical. Our current design for claytronics will not support this scale of an ensemble with this level of fidelity, for example. As the scenario continues, we illustrate what we expect to be nearer-term applications of pario and claytronics.

Now that the "face-to-face" meeting has begun, Alice, Bob, and Charlie get to work. Using MRI or ultrasound, the 3D structure of Fred's injured chest has been scanned. While Fred is resting in a hospital bed in yet another room, a pario model of his chest is rendered using more claytronics material in each of the three rooms. The three doctors interactively explore this physical model to understand the injury: they "peel away" layers on the outside of the model to expose the internal organs, they "zoom in" or "zoom out" (that is, cause the model to grow or shrink), they watch the flow of blood (highlighted by various colors in the claytronics material) through the circulatory system in the chest. While this is happening, the doctors are talking to each other, looking at each other and each change to the model of Fred's chest is synchronized across the rooms: it is as though they are standing in the same room, interacting with the same physical 3D model of the patient.

The interactive 3D physical model application we just described has a different set of requirements and benefits. Unlike the face-to-face example, the claytronics material does not need to be

mobile, requires lower adhesion forces, and can move significantly slower, and has fewer constraints on what acceptable motion will be required.

The initial geometry and appearance of the object (in this case, Fred's chest) do not necessarily come from a camera array, but may come from other arbitrary sources; while we used MRI or ultrasound as the source of the 3D geometry in this example, the information might also come from a CAD program, a scientific computing simulation, and so on. (See figure 9 for an example of how we currently, in simulation, render 3D objects.) To enable hands-on interaction with the physical model, the claytronics hardware must be able to detect touch or pressure to understand how the user is manipulating the object. Unlike a touch-screen device, where only the image on the screen can change in response to human touch, the shape and other physical properties of the object (for example, strength of resistive forces, tactile properties, and so on) can change in an application-specific way: such as "peeling away" particular layers of tissue or bone (by moving them out of the way) in the case of Fred's chest model. Interactive 3D models are compelling in their own right, and they may be used without the face-to-face meeting aspect of the scenario.

> The doctors realize that they also want to consult with Diane (a heart surgeon), who is currently hiking on a remote trail in the woods. Because Diane is in a remote area and she packed light, she does not have enough claytronics material with her for full-blown pario-conferencing. However, Diane does have a small amount of claytronics with her in her pocket (it is currently shaped like a cellphone). Alice, Bob, and Charlie call Diane; they upload the model of Fred's beating heart into her claytronics, which renders itself into a scaled-down version of the same model that Alice, Bob, and Charlie are currently holding in their hands. As Diane manipulates her model with her hands, the same changes are immediately reflected in the other models, and the doctors agree on their treatment plan for Fred. Some of these manipulations are even captured directly by software and later used to guide the actual operation on Fred.

This final aspect of the scenario illustrates how the fact that claytronics can morph between shapes (for example, from a cellphone to a scaled-down model of a patient's chest) may be very useful in a mobile environment. There are many variations on this basic idea, including objects that morph between a shape that is convenient to carry and a shape with a large display, antennas that change their own shape to dynamically adapt the transmission and reception properties, and perhaps even small ingested medical devices that can adapt how they sense and actuate as they move through the digestive tract. In these latter cases, useful devices might be constructed at smaller physical scales, requiring fewer catoms (that is, at a scale of an ensemble that is compatible with our current claytronics design).

Conclusions

As the application scenarios illustrate, pario may fundamentally change how we communicate with others and interact with the world around us. Our research results to date suggest that there is a viable path to implementing both the hardware and software necessary for claytronics, which is a form of programmable matter that can be used to implement pario. While we have made significant progress, there is still much research ahead in order to turn this vision into reality.

While the realization of pario would enable exciting new applications, we also believe that continued research on claytronics itself may offer side benefits to fields such as robotics and machine learning because of scale: we envision millions of (tiny) robots working together to perform a single task (that is, physically rendering an arbitrary moving 3D object), and this is well beyond the scale of typical distributed robotics applications. Hence the techniques that will need to be developed for effectively sensing, reasoning, planning, and actuating on this very large scale to make claytronics and pario work may also be beneficial to future distributed robotics or other AI applications involving very large numbers of cooperating nodes.

Acknowledgements

This research was sponsored in part by the National Science Foundation under grant no. CNS-0428738, Intel Research Pittsburgh, Carnegie Mellon University, and Microsoft Research. The authors would like to thank the reviewers for their excellent comments.

Notes

1. From the Latin root *par,* "to make, to create."

2. Ideally, we would like to construct arbitrary shapes, but this is too stringent a requirement given that the individual units of the ensemble are not made of the same atoms as the object they are rendering. For example, they may not be able to form a shape made from titanium, since the bonds between titanium atoms will allow for moment arms that will exceed those that can be formed between the units.

3. Haptic input devices do not provide the full range of physical interactivity necessary for pario applications.

4. See Stratasys, www.stratasys.com.

5. For more information, see the webcast by Justin Rattner "Research and Development: Crossing the Chasm between Humans and Machines," Day 3 Keynote, Thursday, August 21, 2008, on Intel's website (www.intel.com/pressroom/kits/events/idffall_2008/video.htm).

6. See www.arm.com/products/CPUs/ARM7TDMIS.html.

7. See Panasonic's web page, Multilayer Ceramic Capacitors (for general usage), specifically Ecjzeb0j224m. (industrial.panasonic.com/www-data/pdf/ABJ0000/ABJ0000CE1.pdf).

8. See www.pittsburgh.intel-research.net/dprweb/.

9. See also Milimeter Scale Catoms (www.cs.cmu.edu/~claytronics/hardware/milliscale.html).

References

Abelson, H.; Weiss, R.; Coore, D. A. D.; Hanson, C.; Homsy, G.; Thomas F.; Knight, J.; Nagpal, R.; Rauch, E.; and Sussman, G. J. 2000. Amorphous Computing. *Communications of the ACM* 43(4): 74–82 (www.swiss.ai.mit.edu/projects/amorphous/).

Ashley-Rollman, M. P.; De Rosa, M.; Srinivasa, S. S.; Pillai, P.; Goldstein, S. C.; and Campbell, J. D. 2007a. Declarative Programming for Modular Robots. Paper presented at

the Workshop on Self-Reconfigurable Robots/Systems and Applications at IROS '07, Friday, 2 November, San Diego, CA.

Ashley-Rollman, M. P.; Goldstein, S. C.; Lee, P.; Mowry, T. C.; and Pillai, P. 2007b. Meld: A Declarative Approach to Programming Ensembles. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Campbell, J. D.; Pillai, P.; and Goldstein, S. C. 2005. The Robot Is the Tether: Active, Adaptive Power Routing for Modular Robots with Unary Inter-Robot Connectors, 4108–15. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Cannon, B.; Hoburg, J.; Stancil, D.; and Goldstein, S. 2008. Magnetic Resonant Coupling as a Potential Means for Wireless Power Transfer to Multiple Small Receivers. Technical Report CMU-CS-08-167, Carnegie Mellon University, Pittsburgh, PA.

Ceri, S.; Gottlob, G.; and Tanca, L. 1989. What You Always Wanted to Know about Datalog (and Never Dared to Ask). *IEEE Transactions on Knowledge and Data Engineering* 1(1): 146–166.

De Rosa, M.; Goldstein, S. C.; Lee, P.; Campbell, J. D.; and Pillai, P. 2006. Scalable Shape Sculpting Via Hole Motion: Motion Planning in Lattice-Constrained Module Robots. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA '06)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

De Rosa, M.; Goldstein, S. C.; Lee, P.; Campbell, J. D.; Pillai, P.; and Mowry, T. C. 2007. Distributed Watchpoints: Debugging Large Multirobot Systems. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA '07)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

De Rosa, M.; Goldstein, S. C.; Lee, P.; Campbell, J. D.; and Pillai, P. 2008. Programming Modular Robots with Locally Distributed Predicates. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA '08)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Dewey, D.; Srinivasa, S. S.; Ashley-Rollman, M. P.; De Rosa, M.; Pillai, P.; Mowry, T. C.; Campbell, J. D.; and Goldstein, S. C. 2008. Generalizing Metamodules to Simplify Planning in Modular Robotic Systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '08)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Endy, D. 2005. Foundations for Engineering Biology. *Nature* 438 (24 November): 449–53.

Funiak, S.; Pillai, P.; Ashley-Rollman, M. P.; Campbell, J. D.; and Goldstein, S. C. 2008. Distributed Localization of Modular Robot Ensembles. In *Proceedings of Robotics: Science and Systems*. Cambridge, MA: The MIT Press.

Goldstein, S. C.; Campbell, J. D.; and Mowry, T. C. 2005. Programmable Matter. *IEEE Computer* 38(6): 99–101.

Jones, C., and Matariae, M. 2004. From Local to Global Behavior in Intelligent Self-Assembly. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA '04)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Karagozler, M. E. 2007. Harnessing Capacitance for Inter-Robot Latching, Communication, and Power Transfer. Master's thesis, Carnegie Mellon University Department of Electrical and Computer Engineering. Pittsburgh, PA.

Karagozler, M. E.; Campbell, J. D.; Fedder, G. K.; Goldstein, S. C.; Weller, M. P.; and Yoon, B. W. 2007. Electrostatic Latching for Inter-Module Adhesion, Power Transfer, and Communication in Modular Robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Kirby, B.; Aksak, B.; Goldstein, S. C.; Hoburg, J. F.; Mowry, T. C.; and Pillai, P. 2007. A Modular Robotic System Using Magnetic Force Effectors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Klatzky, R. L., and Lederman, S. J. 2006. The Perceived Roughness of Resistive Virtual Textures: I. Rendering by a Force-Feedback Mouse. *ACM Transactions on Applied Perceptron (TAP)* 3(1): 1–14.

Klavins, E.; Ghrist, R.; and Lipsky, J. 2006. A Grammatical Approach to Self-Organizing Robotic Systems. *IEEE Transactions on Automatic Control*. 51(6): 949.

Kurs, A.; Karalis, A.; Moffatt, R.; Joannopoulos, J. D.; Fisher, P.; and Soljacic, M. 2007. Wireless Power Transfer Via Strongly Coupled Magnetic Resonances. *Science* 317(5834): 83–6.

Lederman, S. J.; Klatzky, R. L.; Tong, C.; and Hamilton, C. 2006. The Perceived Roughness of Resistive Virtual Textures: II. Effects of Varying Viscosity with a Force-Feedback Device. *ACM Transactions on Applied Perceptron (TAP)* 3(1): 15–30.

Loo, B.; Condie, T.; Garofalakis, M.; Gay, D.; Hellerstein, J. M.; Maniatis, P.; Ramakrishnan, R.; Roscoe, T.; and Stoica, I. 2006. Declarative Networking: Language, Execution and Optimization. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data,* 97–108. New York: ACM Press.

Madou, M. J. 2002. *Fundamentals of Microfabrication: The Science of Miniaturization*. Boca Raton, FL: CRC Press.

Murphy, M.; Aksak, B.; and Sitti, M. 2007a. Adhesion and Anisotropic Friction Enhancements of Angled Heterogeneous Micro-Fiber Arrays with Spherical and Spatula Tips. *Journal of Adhesion Science and Technology* 21(12–13): 1281–1296.

Murphy, M.; Aksak, B.; and Sitti, M. 2007b. Adhesion Behavior of Vertical and Aligned Polymer Microfibers. In *Proceedings of Adhesion Society Symposium*. Blacksburg, VA: Adhesion Society.

Nagpal, R. 2001. Programmable Self-Assembly: Constructing Global Shape Using Biologically-Inspired Local Interactions and Origami Mathematics. Ph.D. Dissertation, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science. Cambridge, MA.

Pfenning, F., and Schurmann, C. 1999. System Description: Twelf—A Beta-Logical Framework for Deductive Systems. In *Proceedings of the 16th International Conference on Automated Deduction (CADE-16),* 202–206. Albany, NY: International Conferences on Automated Deduction, Inc.

Reid, J. R.; Vasilyev, V. S.; and Webster, R. T. 2008. Building Micro-Robots: A Path to Sub-mm3 Autonomous Systems. In *Technical Proceedings of the 2008 Nanotechnology Conference and Trade Show,* Volume 1-3. Cambridge, MA: Nano Science and Technology Institute.

Stoy, K., and Nagpal, R. 2004. Self-Reconfiguration Using Directed Growth. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*. Berlin: Springer.

Vasilyev, V. S.; Reid, J. R.; and Webster, R. T. 2008. Microfabrication of Si/SIo2-Spherical Shells as a Path to Sub-mm3 Autonomous Robotic Systems. Paper presented at the Materials Research Society Fall Meeting, 1–5 December, Boston, MA.

Yim, M.; Shen, W.-M.; Salemi, B.; Rus, D.; Moll, M.; Lipson, H.; Klavins, E.; and Chirikjian, G. 2007. Modular Self-Reconfigurable Robot Systems (Grand Challenges of Robotics). *IEEE Robotics and Automation Magazine* 14(1): 43–52.

Zykov, V., and Lipson, H. 2006. Fluidic Stochastic Modular Robotics: Revisiting the System Design. Paper presented at the

Robotics, Science, and Systems Workshop on Self-Reconfigurable Modular Robots, 19 August, Philadelphia, PA.

Seth Copen Goldstein is a professor of computer science at Carnegie Mellon University, Pittsburgh, PA. His research interests include computer architecture, compilers, reconfigurable computing, programmable matter, and systems nanotechnology. He received his BSE degree in 1985 from Princeton University and his Ph.D. degree from the University of California at Berkeley in 1997.

Todd C. Mowry is a professor in the Computer Science Department at Carnegie Mellon University. After receiving his Ph.D. from Stanford University in 1994, Mowry was on the faculty at the University of Toronto before joining Carnegie Mellon in 1997. He also served as the director of the Intel Research Pittsburgh lab from 2004 through 2007. Mowry's research interests span the areas of computer systems, database performance, and modular robotics. He is a member of the editorial board for ACM Transactions of Computer Systems.

Jason Campbell is a research scientist at Intel Research Pittsburgh, where his research interests include self-reconfigurable modular robotics, autonomous systems, embedded computing, and computer vision. He previously served as a senior executive in several startup companies, including chief technical officer of Zack Systems and vice president, technology at Strategic Economic Decisions. He holds a degree in electrical engineering from Stanford University.

Michael Ashley-Rollman is a graduate student in the Computer Science Department of Carnegie Mellon University. His research interests lie in the area of programming languages. Currently he is focusing on approaches for programming massively distributed systems, such as those envisioned by the Claytronics Project.

Michael De Rosa is a Ph.D. candidate at Carnegie Mellon University, and a member of the claytronics research group. His research interests include motion planning for modular robots, distributed debugging, and high-level languages. He is currently involved in developing LDP, a declarative language for programming and reasoning in distributed systems.

Stanislav Funiak is a graduate student in the Robotics Institute at Carnegie Mellon University. His research interests lie in algorithms for distributed probabilistic inference and machine learning. He is currently focusing on localization in large-scale distributed systems, such as modular robots and camera networks, and on designing robust approaches for distributed collaborative filtering.

James F. Hoburg is a professor of electrical and computer engineering at Carnegie Mellon University. His research interests are in a wide range of applications of engineering electromagnetics, including electric and magnetic field analysis and modeling, magnetic shielding, discrete and continuum electromechanical systems, applied electrostatics, electrohydrodynamics and microfluidics. He received a BSEE degree in 1969 from Drexel University, an SM and EE in 1971, and a Ph.D. in 1975 from the Massachusetts Institute of Technology. Hoburg received the Ladd Award for excellence in research in 1979 and the Ryan Award for excellence in undergraduate teaching in 1980, both at Carnegie-Mellon. He is an elected Fellow of IEEE.

Mustafa Emre Karagozler is a Ph.D student in the Department of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, PA. His research interests are microelectromechanical systems, microrobotics, actuator and sensor design for microsystems. He holds a BS degree in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, and an MS in electrical and computer engineering from Carnegie Mellon University.

Brian Kirby is a research associate at Carnegie Mellon University specializing in hardware system and printed circuit board design. He received his BS in electrical and computer engineering from Carnegie Mellon University.

Peter Lee is a professor and head of the Computer Science Department at Carnegie Mellon University. His research contributions lie in areas related to the foundations of software reliability, program analysis, security, and language design. A Fellow of the Association for Computing Machinery and chair of the board of directors of the Computing Research Association, Peter Lee is called upon in diverse venues as a contributor in research, education, and policy making.

Padmanabhan (Babu) Pillai has been a research scientist at Intel Research Pittsburgh since 2003, where he has been working on a variety of distributed systems projects. His research interests span a variety of systems-related fields, including parallel computing, low-power and power-aware systems, operating systems, and embedded, real-time systems. Babu received a BS in electrical and computer engineering from Carnegie Mellon University, and an MS and Ph.D in computer science from the University of Michigan.

Rob Reid is a research scientist with the Air Force Research Laboratory, Sensors Directorate at Hanscom Air Force Base, MA, where he develops and leads programs applying microelectromechanical systems technology to the development of reconfigurable radio systems. Reid's current programs include building fully autonomous robots at the subcubic millimeter scale, developing high-performance, high-power tunable filters, and implementing components for highly integrated millimeter-wave front ends. In 2007, he received the Sensors Directorate Samuel L. Burka award for his in-house research efforts.

Daniel D. Stancil is a professor of electrical and computer engineering at Carnegie Mellon University. He received his Ph.D from the Massachusetts Institute of Technology in 1981. He has served as associate department head and as associate dean for academic affairs in the College of Engineering. Stancil is an IEEE Fellow and a past president of the IEEE Magnetics Society. His research interests include wireless communications, antennas, and applied optics.

Michael Philetus Weller is a Ph.D. candidate in the Computational Design Laboratory at the Carnegie Mellon School of Architecture. His background in architecture and the philosophy of cognition has led to an interest in the ways that the designed objects that comprise our physical environment shape our society. His research currently focuses on using rapid prototyping, robotics and embedded computation to create artifacts that support novel modes of interaction.