Agent Warp Engine: Formula Based Shape Warping for Networked Applications

Alexander Repenning
AgentSheets, Inc.
6560 Gunpark Dr. Suite D
Boulder, CO 80301
+1 303 530-1773
alexander@agentsheets.com

Andri Ioannidou
AgentSheets, Inc.
6560 Gunpark Dr. Suite D
Boulder, CO 80301
+1 303 530-1773
andri@agentsheets.com

ABSTRACT
Computer visualization and networking have advanced dramatically over the last few years, partially driven by the exploding video game market. 3D hardware acceleration has reached the point where even low power handheld computers can render and animate complex 3D graphics efficiently. At the same time, networking has become ubiquitous. Unfortunately, end-user computing does not yet provide the necessary tools and conceptual frameworks to let end-user developers access these technologies and build their own networked interactive 2D and 3D applications such as rich visualizations, animations and simulations. In this paper we introduce the Agent Warp Engine (AWE), a formula-based shape-warping framework that combines end-user visualization and end-user networking. AWE is a spreadsheet-inspired framework based on variables that can easily be shared among networked clients. To build rich visualizations, end users define these variables, relate them through spreadsheet-like equations and connect them to 2D and 3D shapes. In addition to basic shape control such as rotation, size, and location, AWE enables the creation of rich shape warping visualizations. We motivate the AWE approach with an educational application called Mr. Vetro, a human physiology simulation employing end-user visualizations to convey complex heart, skeleton, and lung interactions and end-user networking for collaborative learning.

Categories and Subject Descriptors
C.2.4 [Distributed Systems]: distributed applications, I.3.5 [Computational Geometry and Object Modeling]: Hierarchy and geometric transformations

General Terms
Design, Human Factors, Languages

Keywords
Real-time Image Warping, Collective Simulations, 3D Graphics, spreadsheets, End-User Programming, End-User Development.

AVI'08, May 28-–30, 2008, Napoli, Italy.

1. INTRODUCTION
End-user computing, including end-user development [9] and end-user programming [13], is a quickly growing field with the number of end-user programmers already exceeding the number of professional software developers [7]. Some end-user development employs Web 2.0[1] frameworks, mostly for collaborative authoring and access to shared repositories of non-computational artifacts like images (Flikr), text (Wikipedia), and movies (YouTube).

End-user programming goes beyond the authoring of images, text, and movies by letting computer users without formal programming background create computational artifacts. Spreadsheets have been an extremely popular end-user programming platform. In the 1990s we witnessed a wealth of research that pushed the boundaries of the spreadsheet paradigm (e.g., Forms/3 [3], NoPumpG [20], Garnet [12], AgentSheets [14]). Most of these systems explored extending the existing number-and-string spreadsheet framework with notions of interactive graphics.

Video games and the Web have been essential drivers of the incredibly rapid evolution of personal computers. Since the 1990s, visualization and networking capabilities of affordable computers have exploded, yet very little of these advancements are accessible to end-user computing. Although spreadsheets use 3D technology such as OpenGL to display pie charts and 3D plots very efficiently, from a conceptual viewpoint the state of the art has not changed much.

Perhaps more important than the availability of powerful technology, are the concrete needs we have encountered for a new end-user framework that is capable of creating sophisticated, interactive, networked visualizations. Educators presented us with the challenge of teaching about interacting complex systems such as human systems in physiology. As a response, we created a framework that goes beyond regular animations to create medical visualizations and networked simulations [16]. This framework is based on spreadsheets ideas in a way that provides what we call rich *end-user visualizations* and *end-user networking,* with the following requirements.

[1] http://en.wikipedia.org/wiki/Web_2

End-User Visualizations:

- ***End-User Accessible***: End users should not only be able to select from menus of preexisting visualizations such as plots, and simple geometric shapes, they should also be empowered to construct their own visualizations. In order to do this, they need to be able to make or import 2D or even 3D shapes and have the means to control these shapes so they can serve as visualization.
- ***Rich***. To be truly engaging, visualizations need to be rich. Many variables, e.g., physiological variables like heart rate, could be represented by numbers. However, to truly engage learners, crucial variables should be able to immerse learners audio-visually. Ideally, this would go so far as to be able to even invoke an emotional response. A set of variables representing a hyperventilating human being should not just be a list of numbers, but should show a human being with heart and lung functions – even with sounds that provoke emotional responses.
- ***Efficient***. To be perceived as smoothly animated, visualizations need to be highly efficient. In the case of Mr. Vetro we need to be able to perform complex shape warping for the skeleton, heart, and lung in real time and at a frame rate of at least 30 frames per second.

End-User Networking:

- ***Transparent networking***: Similar to a spreadsheet, it should be possible to define computation as a set of variables connected with each other through formulae. Unlike the traditional spreadsheet idea, however, variables should be sharable through the Web. This mechanism should be as transparent as possible. That is, an end user should not have to use network APIs or use any other complex mechanism to implement variable sharing. All they should have to do would be to declare the variable to be shared when they create that variable.
- ***Real time computation***: To build sophisticated animations and simulations it should include a simple model of time similar to the Forms/3 system [3]. Additionally, because of potential delays caused by networking and various client hardware and software implementations, animations, and simulations need to be able to work in a frame rate independent way.
- ***Fluid***: End-user networking should exhibit low viscosity [6]. Similar to a spreadsheet, it should be simple for the user to add, change, and remove variables. The environment should facilitate experimental explorations of networked computing.

In the following sections we introduce the Agent Warp Engine technology with a basic example, describe Mr. Vetro as a more sophisticated medical application, compare related work, and share our evaluation results.

2. TECHNOLOGY

The Agent Warp Engine (AWE) is a technical framework for running collective simulations. It is implemented as thin layer on top of the Open Agent Engine[2] (OAE) which itself is the open source part of the AgentCubes 3D game and simulation-authoring environment [15]. The OAE implements a simple 3D agent-based simulation engine based on four main components:

OpenGL: ***2D/3D Graphics.*** A highly optimized 2D/3D rendering API[3]. Fundamental graphics primitives such as 3D meshes, textures, and shaders [19] are hardware accelerated on most platforms. OpenGL is used to render large numbers of agents with 3D shapes efficiently.

QuickTime: Sounds, images, movies. An API available for OS X, Windows, and Linux[4] provides access to image, movie, and sound files. QuickTime is used to load texture files and to play sound files.

XMLisp: files, knowledge representations. XMLisp [17] is an API mapping XML expressions to object oriented language constructs. An AWE project consists of XML files that are processed through XMLisp.

3D Agents. Autonomous objects that have a 3D position, orientation, size, velocity, and acceleration. Agents can be composed into scenes, animated, displayed and user selected.

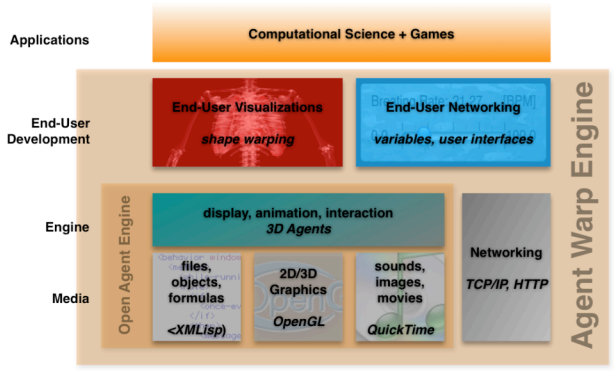

Figure 1. The Agent Warp Engine is layered on top of the Open Agent Engine

AWE adds two main components to the architecture (Figure 1):

- ***End-User Visualization***: end users create custom visualizations by defining 2D or 3D shapes with control points that connect to variables through spreadsheet-like formulas. Employing techniques such as shape warping, users can define sophisticated visualizations such as a beating human heart.
- ***End-User Networking****:* an end-user formula language includes the notion of variables and equations connecting them. Variables on different clients can be shared via the network in real time. Network access, established via HTTP, is completely transparent to users.

The following two sections illustrate the notions of end-user visualization and end-user networking in detail.

2.1 End-User Visualization

An important goal of this work is to offer the refined kinds of visualizations necessary to communicate complex dynamic processes. For instance, in our main application called Mr. Vetro, we need to show the function of the heart, the lung, and the

[2] http://www.agentsheets.com/lisp/OpenGL.html

[3] http://www.opengl.org

[4] http://www.openquicktime.org

human skeleton. All three systems mechanically interact with each other in complex ways. For instance, inhaling air will change the shape of the lung, which in turn will influence the skeleton. Rips expand and, in the case of deep breathing, even the position of the shoulders and arms can be influenced. AWE offers a number of visualizations, but the most sophisticated one (called "morph") is specifically designed to implement complex visualization based on shape warping. Shape warping is a kind of image warping [21, 22].

We start with a basic but detailed example applying a shape warp visualization to the well known image of Leonardo DaVinci's Mona Lisa. While this example is much simpler than the Mr. Vetro visualization described in the next section, it illustrates how even small, easy to program shape warps can have a major effect. It helps that the human perception system is highly sensitive towards emotional clues found in facial expressions.

Experts are still debating if Mona Lisa's facial expression represents sadness or happiness. We will not contribute to this discussion, but we will use Mona Lisa's image to illustrate how to build a simple visualization with AWE. We will use morphs as means to warp Mona Lisa's image in a controlled way and evoke different emotional interpretations.

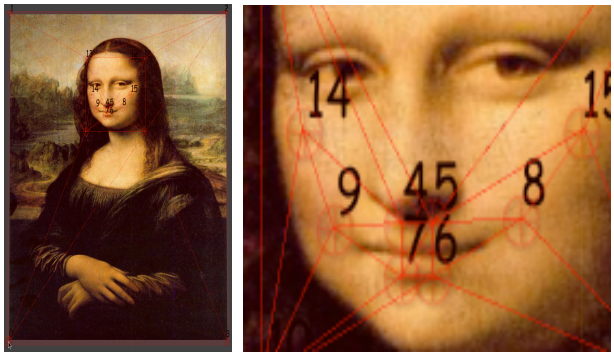

Figure 2: Left: Mona Lisa. Right: detail showing tessellated detail of her face.

An AWE morph can warp images using texture mapping [21]. Explaining the details of texture mapping goes beyond the scope of this paper, but in essence, a texture map is a means of applying a texture or image to a shape. A simple example would be to map a rectangular image such as the entire image of the Mona Lisa onto a rectangular shape. The pixels of the image need to be mapped accordingly, e.g., scaled vertically or horizontally depending on the shape dimensions. Things become substantially more interesting when the target shape is deformed in ways other than just vertical and horizontal scaling. For instance, one can tessellate an image, a process of tiling, by segmenting it into a triangle mesh. Mapping these triangles onto identical target triangles would create the original picture. However, one can warp an image by shifting vertices of this mesh. To render a warped image, texture-mapping needs to be able to warp individual triangles in terms of shape and their image content. This process is computationally intensive but can be done very efficiently in hardware on modern graphical processing units.

The main idea of our simple emotional visualization example is to add a single variable to control the entire spectrum from sad ☹, to neutral 😐, and on to happy ☺. Sophisticated facial expression systems have existed for some time. The Radial Basis Function network approach [8], for instance, uses 2D image warping based on neural nets. Here we want to show a much simpler approach that is readily accessible to end users.

The first step in defining our visualization is the image tessellation. AWE includes a mesh-authoring tool that lets end users define tessellation points and triangles. A simple approach to warping our image emotionally is to focus on Mona Lisa's mouth to make her look happy, sad or neutral. A mesh around her mouth (Figure 2, vertices 4, 5, 6, 7, 8, 9) is a starting point. Additional vertices are needed to be able to define triangles covering the entire image. The selection of vertices requires some experience. Users participating in our evaluation (section 5.2) were able to create a well working mesh of a person that included warping of mouth and eyes after seeing one example mesh. The key to vertex selection is controlling the scope of the desired effect. To change the mouth by moving vertices 4-9, one needs to make sure that the mouth deformation does not influence too much of the remaining image. For instance, if the only other vertices were the corners of the image itself, then moving vertices 8 and 9 up to make Mona Lisa smile would also partially move the rest of the face in an unnatural way. Instead, we define vertices 14 and 15 as fix points, roughly at the location of the cheekbones.

To create a *formula-based shape warp* an end user needs to define a mesh and add formulas to vertices. The AWE mesh-authoring tool automatically creates an XML representation of the Mona Lisa mesh that includes the image reference, a list of vertices, and a list of triangles. Vertices 8 and 9 are the left and right corners of the mouth. Vertex 8 in AWE XML notation looks like this:

```
<vertex x="0.520" y="0.712" xt="0.520" yt="0.712"/>
```

X and Y are the vertex positions, whereas xt and yt are the texture coordinates.

The goal is to control Mona Lisa's emotions by adjusting the positions of the left and the right corners of the mouth. Both the x and the y attribute of the vertex are extended by the user from being constants to being formulas:

```
<vertex x="0.520 + 0.0003 * happiness" y="0.712 + 0.0003 *
   happiness" … />
```

Happiness is a user defined variable. For happiness = 0 we get the original image. For happiness > 0 we get an increasingly happy Mona Lisa by pulling her mouth corners up and out. Finally, for happiness < 0 we have her start to frown by pulling her mouth corners down and together.

The real power of a formula-based shape warp appears when the user sees it attached to a variable controlled by a slider and experiences warping in real time. Because of OpenGL hardware acceleration, this simple application with both visualization and formula evaluation runs at hundreds of frames per second—even on moderate hardware. Unfortunately, this experience cannot be completely shared in a static publication. However, Figure 3 shows some variations—including the original image. With this simple mesh and equations used in this example, the limits are visible. For instance, the super happy Mona Lisa no longer looks completely natural. However, the important point is that end users with no background in computer graphics can build formula-based shape warps effectively.

2.2 End–User Networking

End-user networking is based on shared variables that let end users create distributed applications. End users should be able to define, change, and connect networked variables as easily as they

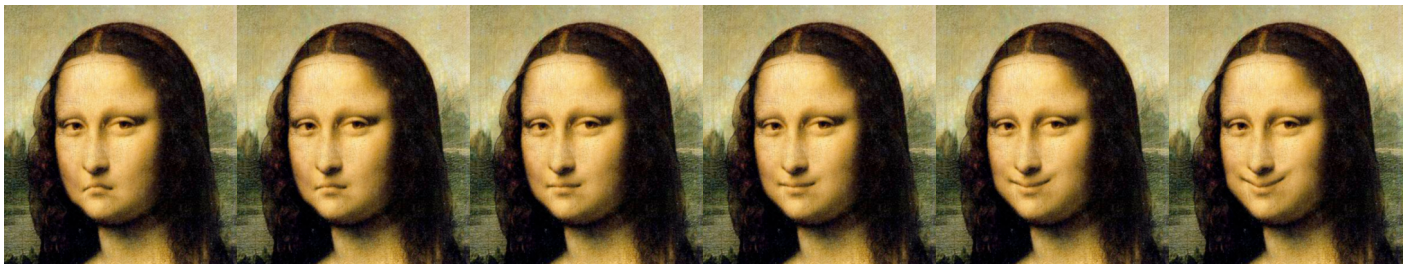

Figure 3: Shape warping of Mona Lisa. Depending on Happiness she is annoyed, concerned, innocent, enigmatic, happy, ecstatic.

use variables in spreadsheets. For our collective simulations applications, it is essential that these variables can be shared on different clients running on networked computers.

For most of our applications, real-time interactivity is essential. In collaborative simulation all users are physically present in the same room. Being able, as a group, to see how individual users change simulation variables in real time and how the system reacts to change helps the perception of causality [11]. Users typically change the value of variables gradually using slider interfaces to let themselves and everybody else in the classroom experience controlled change and the gradual reactive effect of all the other simulations that depend on a specific variable.

AWE supports *local variables* and *shared variables*. All variables have these properties:

- ***name***: all variable references are by user-defined names.
- ***value***: a value is defined as constant (e.g., value=3.14), a formula (e.g., value="sin(time) + 45.0") or through a user interface such as a slider (Figure 4).
- ***action***: each time the variable is changed an action could be invoked.
- ***user interface***: one or more user interfaces can be attached to a variable to output values or to get values from the user.

Local variables are the simplest kind of variables. They could be compared to cells of a spreadsheet. The scope of local variables is limited to a single AWE client. Other AWE clients may have local variables with identical names without establishing any kind of sharing.

Shared variables are shared through the Web. A user simply defines a variable to be shared without the need to explicitly deal with network interfaces. For instance, if the variable heart_rate is defined to be a shared variable, then all clients can simply access that variable by referring to its name. For instance, the equation `sin(time * heart_rate * 6.28 / 60)` will access the current heart_rate.

One client writes a shared variable, but any number of clients can read a shared variable. Shared variables are shadowed by local variables with the same name.

The sharing of shared variables is achieved through a web server. Clients establish fast HTTP 1.1 persistent connections to a shared web server that is storing and providing access to variable <name, value> tuples. Each client has a number of caches holding values of local and shared variables in order to minimize networking overhead. When a formula is evaluated, all its variable values are computed. If the value is not found in the right cache then it will be assumed that the variable to be accesses is a shared variable.

User interfaces allow users to control the values of variables. Mona Lisa's shape warp employs a formula that includes a single variable reference: `0.520 + 0.0003 * happiness`. Happiness was defined to be a local variable with a slider user interface. The XML representation

```
<local-variable name="HAPPINESS">
  <slider-interface min-value="-50.0" max-value="50.0"
    label="happy:" units="[haha]"/>
</local-variable>
```

corresponds to the slider in Figure 4, representing a local variable. As the user changes the value of the variable through the slider, the shape warp is recomputed and updated on the screen. Displaying the slider and the shape warp is fast; they render at about 400 frames per second on a 1.67 Ghz Mac PowerBook G4 with an ATI Mobility 9700 GPU.

Figure 4. Variable Happiness with slider user interface

User interface output options include sound. In the Mr. Vetro application (Figure 5) the lung distortion is computed as a function of time, breathing rate, and lung tidal volume. To increase the immersiveness of the visualization we added inhale and exhale sounds that are triggered if the value of the distortion variable begins to increase or to decrease respectively.

```
<local-variable name="DISTORTION" value="0.02 *
  lung_tidal_volume * sin(time * breathing_rate * 6.28 /
  60)">
  <sound-alert-when-value-begins-to-increase
    soundfile="inhale.mp3"/>
  <sound-alert-when-value-begins-to-decrease
    soundfile="exhale.mp3"/>
</local-variable>
```

Next we describe the Mr. Vetro application in more detail. This application was the main driver for developing formula based shape warping.

3. MR. VETRO: AN AWE APPLICATION

Mr. Vetro[5,6] (Figure 5) is a Collective Simulation for human physiology we have developed in collaboration with teachers and medical doctors for use in K-12 science classes [16]. *Collective Simulations* is a conceptual framework that integrates *social learning pedagogical models* with *distributed simulation technical frameworks*. This conceptual framework both enables and actively encourages meaningful learning by supporting a discovery-oriented social learning process. Visualization, animation and

[5] Translated from Italian, "vetro" means "glass". The name is derived from Mr. Vetro's glass skeleton.

[6] An interactive flyer with Mr. Vetro can be found at: http://agentsheets.com/research/c5/documents/interactive%20flier/c5-flier.html.

simulation as well as networking play a vital role in collective simulations. The first incarnation of Mr. Vetro (described below) incorporated these aspects, but our preliminary evaluation (section 5.1) led to making those aspects accessible to end-users. This process led to the development of the end-user visualization and end-user networking components of the AWE architecture.

In the Mr. Vetro application, different human systems are simulated on wirelessly connected handheld computers, while a central simulation aggregates parameters from the organs and computes Mr. Vetro's vital signs.

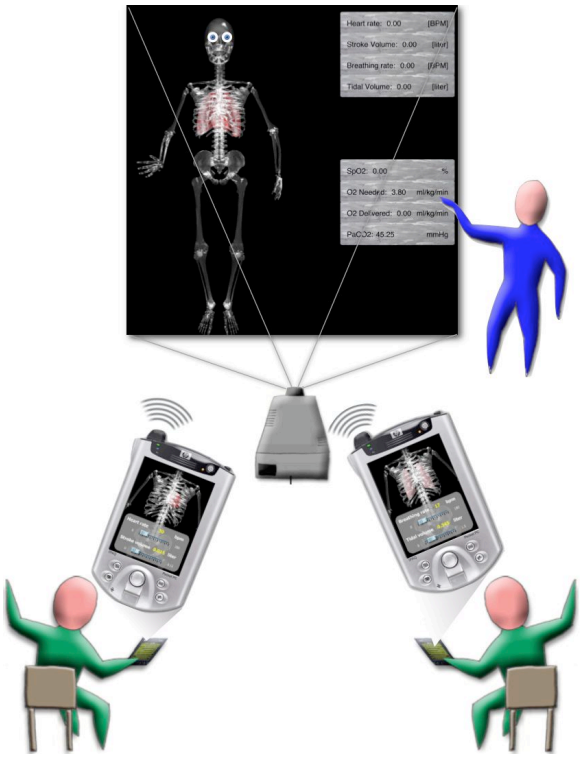

Figure 5: Mr Vetro – the collective simulation aggregating all the input parameters from the distributed organs and calculating vital signs.

In an activity, handheld devices with Mr. Vetro's organs are handed out to students. One group receives the heart, another group the lungs. Students engage in role-playing by being Mr. Vetro's organs. The lung group varies parameters such as breathing rate and tidal volume in response to changing conditions such as exercise or smoking. The heart group can vary parameters such as heart rate and stroke volume. Another group controls decisions such as how intensely to exercise.

A computer connected to a video projector is the server that communicates wirelessly with all the client simulations and aggregates all the inputs into a composite representation of a human that includes visual and audio elements. A vital signs monitor keeps track of Mr. Vetro's vital signs and displays them in the form of graphs or numerical values. Oxygen saturation in the blood, partial pressure of CO_2, oxygen needed and oxygen delivered to tissue are some of the physiological variables that are calculated and displayed.

Using Collective Simulations in the classroom is dramatically different from the typical use of technology in education. A collective simulation cannot be operated without discourse and collaboration among members of the same team and among different teams. This fosters a social style of learning that emphasizes distributed cognition [2].

3.1 Requirements

In a formal feasibility study conducted with the first prototype of Mr. Vetro (section 5.1), we found that collective simulations provide significant improvement over lecture-style teaching. This pilot study provided strong evidence for effective teaching performance and increased motivation. However, this real-world classroom use of the technology also revealed some emerging requirements that are being addressed in the next research phase.

Crude Animations => Improved Medical Content: Animations of the human were crude. Mr. Vetro's skeleton was not moving at all and the lung animation was anatomically wrong. These limitations were pointed out by both medial doctors and students during the evaluation study. Our limited animations needed to be replaced with more anatomically and physiologically correct animations that would be acceptable to medical specialists (doctors & medical illustrators). We therefore designed the AWE engine for end-user visualizations.

Networking challenges => Zero Configuration: The distributed nature of collective simulations presented non-trivial network configuration challenges. Establishing connectivity between clients, servers, and the Web proved to be difficult in educational settings. We found wireless networking to be especially complex for current-generation PocketPC PDAs. In educational settings teachers have no time to spend on elaborate technology configurations. Simplifying this process to the point of zero configuration will be essential to successful adoption of this technology. To simplify connectivity we developed a new approach. Instead of embedding the collective simulation in a server running in the classroom – as in Mr. Vetro I – which required client reconfiguration each time the network environment changed (e.g. typing IP addresses on each client), we designed a client server architecture where everything (individual organs as well as the entire Mr. Vetro) is a client to a Tuple [10] server. The server resides on a machine accessible from anywhere and does not require special configuration for the clients to access it.

Hard-coded system => Flexible authoring mechanisms: Extending or altering the activity to introduce more physiological variables was not easily achievable with the first system prototype. However, easy alteration and system flexibility is needed to extend the repertoire of interactive educational activities. This extension is commonly achieved by customizing existing activities and including new ones that focus on different aspects of human physiology. From the activity developer's point of view the system should have low viscosity (resistance to change) [6]. The effort required to change end-user defined artifacts should not be prohibitive. Instead, flexible mechanisms should let the activity designer create new activities easily. For instance, new organs and many physiological variables could be added to Mr. Vetro. But from the user's perspective, it does not make sense for everything to be exposed at all times. Depending on the educational goals of each scenario, different organs and variables could be involved, but the students would only see a subset. Since human physiology is extremely complex and involves convoluted relationships of many organs and variables, it is often necessary to reduce the complexity to make it manageable for students to infer, and understand relationships between variables.

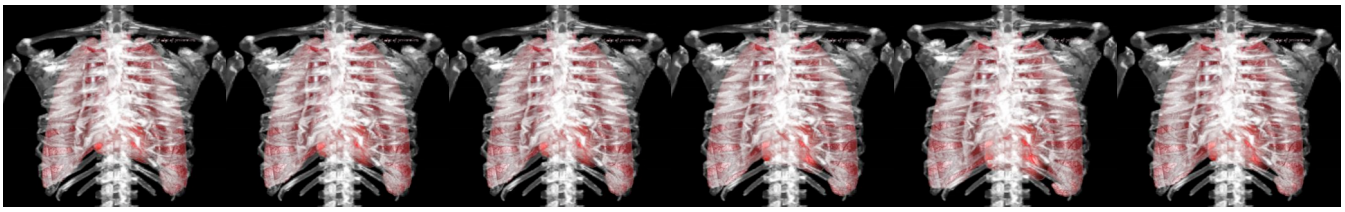

Figure 6: Sequence of frames in Mr. Vetro's animation: the visualization of breathing includes movement of the skeleton (notice the shoulder location changing in each successive frame) and lungs filling up the space at the bottom when the diaphragm recedes to make room for them to expand. The visualization of the heartbeat is illustrated by subtle warping of the heart shape.

3.2 Mr Vetro II

The need to address the needs uncovered by the feasibility study led to the design of the new Agent Warp Engine infrastructure for our collective simulation framework. End-user visualizations tailored to Mr. Vetro provide the complex animations needed for realistic and accurate medical applications. These visualization feature formula-based shape-warping capabilities based on user-defined variables. They also enable flexible authoring of new organs of the human body. End-user networking for collective simulations addresses networking challenges by reconceptualizing the architecture and implementing every organ as a client to a Tuple server. It also enables flexible authoring of new activities with the introduction of new physiological variables.

Mr. Vetro II consists of the following shape warps:

Skeleton: A ray-traced shape with 16,000 polygons is used to create the X-ray look of Mr. Vetro's glass skeleton. The skeleton movement is animated based on a distortion equation that is a function of breathing rate and tidal volume. The effect is a realistic shoulder and ribs movement based on how fast, and how deep Mr. Vetro is breathing (Figure 6).

Heart and Lungs: The classic illustration of the heart and lungs is from Gray's Anatomy book [5]. The organs are both warped independently based on parameters of each organ, and also move synergistically to simulate the mechanical connection between the two organs in the body.

Mr. Vetro includes the following local and shared variables:

Heart parameters: Heart rate and stroke volume are shared variables of the distributed client simulation of the heart.

Lung parameters: Breathing rate and tidal volume are shared variables of the distributed client simulation of the lungs.

Activity parameters: exercise intensity expressed in running speed is a shared variable of the distributed client simulation of Mr. Vetro's brain.

Vital signs: O_2 saturation, partial pressure of CO_2, O_2 needed, and O_2 delivered to tissue as Mr. Vetro exercises are local variables computed based on equations referring to input parameters.

For continuous shape warping (having the animation/warp change in real-time) a special time variable is used in the distortion functions. For instance to simulate the motions of the human body associated with breathing, or the movement and distortions of lungs and heart as the breathing and heart parameters change, we need to express the warping of the vertices common between lung and heart as a function of two different frequencies, as well as time, e.g.:

```
<vertex x="0.735 + distortion_heart" y="0.120 - 4 *
   distortion_lung + distortion_heart" xt="0.735" yt="0.120"/>
```

where distortion_heart is defined as

```
<local-variable name="distortion_heart" value="0.2 *
   heart_stroke_volume * sin(time * heart_rate * 6.28 / 60)" />
```

The sequence of frames in Figure 6 show Mr. Vetro's skeleton, heart and lungs move according to the input parameters. Since it is difficult to capture animation on print media, we created a movie that conveys the full animation experience[7].

4. RELATED WORK

The AWE framework for formula-based shape warping for networked applications is a spreadsheet-inspired approach to end-user visualization and networking.

Since its early inception as a one-way dataflow constraint system, the spreadsheet paradigm evolved to include visualizations. For example, in the ThingLab system [1], a constraint-oriented simulation laboratory was used for the development of interactive graphics. ThingLab supported the definition of networks of constraints, but the bidirectional nature of constraints were hard to predict [20]. Moreover, there was no real end-user component: the non-graphical, programming-language notation for the constraints did not let the user author – there was a clear distinction between the "user" of the system and the programmer [20]. NoPumpG [20], a system that combined interactive graphics with spreadsheets, attempted to introduce more end-user authoring aspects to the definition of visualizations (behavior and appearance). Forms/3 [3] extended visualization capabilities with a notion of time used to create animations and simulations.

Networked spreadsheets expanded the spreadsheet paradigm with networking capabilities. Spreadsheet-inspired tools such as WikiCalc[8], a web application that let users share spreadsheets through wiki-style user-editable interfaces, enabled new kinds of collaboration through end-user networking. However, WikiCalc does not include sophisticated visualizations, and more importantly, is not geared for real-time variable sharing.

The AgentSheets Behavior Exchange [18] was an early incarnation of Web 2.0 ideas combining end-user visualizations with end-user networking. It let end users author and share computational components, called agents, through web interfaces. Used mostly for educational applications, the Behavior Exchange let end users, such as children in science education, learn about the fragility of ecosystems by creating and sharing electronic versions of behaving animals. In contrast to text, photos, and movies, the agents that were created with this kind of technology included end-user defined behavior. Today, we see early versions

[7] Movie: http://www.youtube.com/watch?v=39NJJC1Vt18. Alternatively go to YouTube (http://www.youtube.com) and search for Mr Vetro.

[8] http://en.wikipedia.org/wiki/WikiCalc

of more sophisticated forms of end-user programming that include the use of web services to execute programs useful to end users.

Mashup platforms such as Yahoo Pipes[9] or Orchestr8[10] are an exciting development that bring web-based computing to end users. End users can create their own mashups by running, inspecting, and editing shared scripts. Yahoo Pipes, for example, lets end users define scripts called pipes to access and process web based information available in RSS format.

5. EVALUATION

Evaluation for AWE technology has occurred at multiple levels. The educational efficacy of the collective simulation approach was evaluated in a formal feasibility study in high school biology classes; the usability of the end-user visualization approach of AWE has been evaluated in an undergraduate Computer Graphics and Visualization class; and end-user networking is currently being evaluated as we work with teachers and content experts to create new activities for physiology classes.

5.1 Educational evaluation

To evaluate the feasibility of the Collective Simulation framework with Mr. Vetro, we designed an experiment comparing a traditional lecture format with an interactive activity using the Mr. Vetro collective simulation. We worked extensively with teachers and doctors to create two complete alternative learning activities aligned with state-level science standards to use in the evaluation study. We employed an improvement-measuring framework [4], to assess the advantages of educational technology. The framework looks for evidence of four different types of improvements: *increased learner motivation*, *advanced topics mastered*, *students acting as experts do*, and *better outcomes on standardized tests*.

We considered the feasibility study to be a success if in the collective simulation condition group we measured performed equal to or better than the lecture group, and we saw evidence for increased student motivation, mastery of advanced topics, and ability to act as experts. Along the four evaluation dimensions the results were as follows:

Learner motivation: Motivation is essential and at the core of this research. We strongly believe that without motivation much of the educational effort is lost. Students in the lecture group were not particularly engaged in the material being presented. In comparison, in the Mr. Vetro group, students were noticeably engaged. The teacher asserted that students stayed completely focused on task and more students were engaged speaking, listening, and asking questions than in any lecture session and most lab sessions.

Mastery of advanced topics: Mastery of a topic is not limited to memorizing facts, but includes the ability to gain deeper understanding of knowledge. The Mr. Vetro groups scored significantly higher (26% higher) than the control group on the deep knowledge questions posed on the retention test. In order to answer these questions, students had to have a strong sense of the interaction of human organs as they had to reason quantitatively about physiological variables at settings they had not experienced during either the lecture or the simulation.

Learners acting as experts: Developing the ability in learners to use problem solving processes similar to those of experts is challenging, but provides powerful evidence that students are gaining the skills they will need to succeed. The inquiry-based approach to pedagogy creates a need to work and communicate in groups and substantially changes the roles of students and teachers. Students engage in cause and effect questions. Rather than behaving in the typical student role of absorbing facts and vocabulary, Mr. Vetro students were grappling with ideas and trying to find answers to their own questions. When students expressed a misguided interpretation of the current situation, other students intervened to correct the misconceptions. Students that normally did not participate in class took on leadership roles in their groups.

***Performance on conventional tests*:** The most difficult type of evidence to provide for the superiority of new, technology-based instructional models is higher scores on conventional measures of achievement. There is no standardized test for human physiology at the high-school level. Instead, we used teacher-generated tests that were guided by the biology curriculum and standardized science tests. Comparing performance scores (total points on the retention test) of both groups we found a slight, insignificant ($p < 0.05$) advantage in the collective simulation condition. However, we found that the collective simulation group had a significantly lower pre-test average score compared to the lecture group. Therefore, in terms of learning gain (ratio of retention score to the pre-test score) we found a significant advantage for the Mr. Vetro group: this group showed a 58.88% learning gain compared to a 50.87% learning gain of the lecture group.

This formal feasibility study provided early indications of the educational efficacy of the collective simulations approach for teaching about the relationships of complex interacting systems.

5.2 End-User Visualization evaluation

The idea of formula-based shape warping was evaluated with a group of undergraduate informatics students with no background in computer graphics. In a Computer Graphics and Visualization class at the University of Lugano in Switzerland, the task was to create a shape warp of faces similar to the Mona Lisa example presented in this paper. In addition to making the faces appear to smile or be sad by warping the mouth, students had to warp the eyes. Moreover, the faces had to be able to open and close their eyes without squeezing the eye pupils. This required a more sophisticated approach of having two layers of images. One layer was the image of the person with the eye masked out, and the other layers, behind the first one, were the eyes as two separate images. All students in this experiment were able to create a running version of the face morph application.

5.3 End-User Networking evaluation

We are currently working with biology teachers and doctors to add new systems to Mr. Vetro. Adding the renal system (the kidneys) would allow us to explore scenarios of blood loss, dehydration, and training in high altitude. For instance, Mr. Vetro could be in situations where blood pressure changes dramatically and needs to be regulated. To accomplish this, the kidneys need to produce Renin to start the chain reaction of turning Angiotensin I to Angiotensin II; Angiotensin II ultimately regulates blood pressure. In these scenarios, blood pressure could be an input to the system. In other scenarios, like subjecting Mr. Vetro to exercise, the blood pressure would be an output that is calculated and visualized for the students. The new AWE architecture allows

[9] http://pipes.yahoo.com

[10] http://www.orch8.net

such variations. Two representations of Mr. Vetro could exist for each scenario with different XML representations of the blood pressure variable – one defined as a shared variable with user input interface (e.g. a slider), and one as a calculated output.

6. DISCUSSION

Our current method of developing these activities involves collaborating with content experts to get enough information to produce the code (XML representations) for the distributed components of the collective simulation. Our ultimate goal is to give these experts the tools they need to define the system interactions with AWE by themselves. Most of the XML expressions needed to specify a formula based shape warp is automatically generated with the authoring tool. However, it is still necessary to find the proper locations for relevant vertices and to add formula expressions. Surprisingly, we found that the subjects in our study had few problems editing XML files. In part, we can explain this by end users having an increasing familiarity with XML and HTML file formats. Nonetheless, syntactic challenges due to XML editing can and should be eliminated.

Advances in computer graphics and mobile computing make it possible to create a new kind of networked application with strong visualization, animation and simulation components. The Agent Warp Engine not only combines sophisticated 2D/3D visualizations and real-time networking but also makes them accessible to end users. Formula based shape warping is a spreadsheet inspired end-user programming paradigm that can be employed for a variety of applications in need of end-user visualizations and end-user networking. In addition to the technological framework itself we have presented end-user visualization and end-user networking aspects of the Mr. Vetro human physiology collective simulation.

7. ACKNOWLEDGMENTS

This work is supported by NIH Grant 1R43 RR022008-02. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Institutes of Health.

8. REFERENCES

[1] Borning, A. 1981. The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory. ACM Transactions on Programming Languages and Systems (TOPLAS), 3, 4, 353 - 387.

[2] Brown, A.L., Ash, D., Rutherford, M., Nakagawa, K., Gordon, A. and Campione, J. Distributed Expertise in the Classroom. in Salomon, G. ed. Distributed Cognitions, Cambridge University Press, New York, 1997.

[3] Burnett, M., Atwood, J., Djang, R., Gottfried, H., Reichwein, J. and Yang, S. 2001. Forms/3: A First-Order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm. In Journal of Functional Programming (March), 155-206.

[4] Dede, C. 1996. Emerging technologies and distributed learning. American Journal of Distance Education, 10(2), 4-36.

[5] Gray, H. Anatomy of the Human Body. LEA & Febiger, Philadelphia, 1918.

[6] Green, T.R.G. 1989. Cognitive Dimensions of Notations. In Proceedings of the Fifth Conference of the British Computer Society. Cambridge University Press, Nottingham, 443-460.

[7] Jones, C. 1995. End-user programming. IEEE Computer, 28(9), 68-70.

[8] Leung, M.Y.Y., Hung, Y.H. and King, I. 1996. Facial expression synthesis by radial basis function network and image warping. In Proceedings of the IEEE International Conference on Neural Networks (June 3-6). IEEE, Washington, DC, USA.

[9] Lieberman, H., Paternò, F. and Wulf, V. (eds.). End User Development. Springer, 2006.

[10] Matsuoka, S. and Kawai, S. 1988. Using Tuple Space Communication in Distributed Object-Oriented Languages. In OOPSLA '88. ACM Press, San Diego.

[11] Michotte, A. The perception of causality. Methuen, Andover, MA, 1962.

[12] Myers, B., Hudson, S.E. and Pausch, R. 2000. Past, present, and future of user interface software tools. ACM Transactions Computer-Human Interaction, 7(1), 3-28.

[13] Nardi, B. A Small Matter of Programming. MIT Press, Cambridge, MA, 1993.

[14] Repenning, A. and Ioannidou, A. 2004. Agent-Based End-User Development. Communications of the ACM, 47(9), 43-46.

[15] Repenning, A. and Ioannidou, A. 2006. AgentCubes: Raising the Ceiling of End-User Development in Education through Incremental 3D. In IEEE Symposium on Visual Languages and Human-Centric Computing 2006 (September 4-8). IEEE Press, Brighton, United Kingdom.

[16] Repenning, A. and Ioannidou, A. 2005. Mr. Vetro: A Collective Simulation Framework. In ED-Media 2005, World Conference on Educational Multimedia, Hypermedia & Telecommunications. Association for the Advancement of Computing in Education, Montreal, Canada.

[17] Repenning, A. and Ioannidou, A. 2007. X-expressions in XMLisp: S-expressions and Extensible Markup Language Unite. In Proceedings of the ACM SIGPLAN International Lisp Conference (ILC 2007). ACM Press, Cambridge, England.

[18] Repenning, A., Ioannidou, A., Rausch, M. and Phillips, J. 1998. Using Agents as a Currency of Exchange between End-Users. In Proceedings of the WebNET 98 World Conference of the WW, Internet, and Intranet. Association for the Advancement of Computing in Education, Orlando, FL, 762-767.

[19] Rost, R.J. OpenGL Shading Language (2nd Edition). Addison-Wesley, 2006.

[20] Wilde, N. and Lewis, C. 1990. Spreadsheet-based Interactive Graphics: From Prototype to Tool. In Proceedings CHI'90. ACM Press, Seattle, WA., 153-159.

[21] Wolberg, G. Digital Image Warping. IEEE Computer Society Press, 1994.

[22] Wolberg, G. 1996. Recent Advances in Image Morphing. In Proceedings of the 1996 Conference on Computer Graphics International. IEEE Computer Society, Washington, DC, 64.