Identity SA - an interactive swarm-based animation with a deformed reflection

Prof. T. Unemi, BEng, MEng, DEng.
Department of Information Systems Science, Soka University, Tokyo, Japan.
e-mail: unemi@iss.soka.ac.jp

Dr. D. Bisig, BSc, MSc, PhD.
Artificial Intelligence Laboratory, University of Zurich, Switzerland.
e-mail: dbisig@ifi.unizh.ch

Abstract

Identity SA is an interactive and generative installation that combines a swarm-based simulation with real time camera based interaction. The agents' distributions are transformed into "painterly" images by employing a variety of different visualization techniques and styles, such as texture surfaces, short line segments, font glyphs, curved lines, etc. Camera based Interaction is based on a simple motion detection algorithm that affects the agents' movements as well as their coloring. Normally, an agent's color is solely determined by its orientation, but whenever the tracking system detects a visitor's motion, the agent's color is additionally affected by the corresponding pixel color in the camera frame. It acts as a visual and acoustic mirror, which distorts the continuity of the visitor's physical existence into ephemeral patterns and flowing motions.

1. Introduction

Humans are the only species that strives to achieve an understanding of its nature and purpose. Since ancient ages, we have struggled to find satisfactory answers since our capabilities of introspection are very limited. Often, we try to observe ourselves via a reflection through our social environment. Despite these efforts, our self-image always remains unclear, fuzzy, and ambiguous. We cannot even preclude that it might be an illusion.

Identity SA is a computer based interactive artwork that creates abstract reflections of the visitor's appearance. It acts as a visual and acoustic mirror, which distorts the continuity of the visitor's physical existence into ephemeral patterns and flowing motions. It reminds the visitor of the fragility of his/her own self-awareness and existence. The installation mimics an abstract painting whose moving brush strokes create a portrait of the visitor. The integrity of this portrait can only be preserved by the sustained activity of the visitor. Whenever he/she stops moving, the portrait becomes increasingly vague and fuzzy and eventually disappears entirely in a swirl of colors. This continuous struggle versus one's own disintegration metaphorically reflects our permanent exposure to the treat of losing ourselves through mental and physical decay. According to Buddhist teachings, sickness and aging together with birth and death form the group of four inescapable sufferings [1]. The suffix in the title for this project stands for sickness and aging.

In our former works, Flocking Orchestra [2][3], MediaFlies [4], and Flocking Messengers [5], we have examined combinations of swarm simulation and camera-based visual feedback in order to realize pieces of generative art. Several other artists have chosen similar approaches

in their works [6]. These works employ tens or hundreds of agents moving in a virtual 3D space to play music and/or to draw images. In our newest work that is presented here, we increase the number of agents to thousands but reduce the dimensionality of the agent world to 2D. As the agents move through this flat canvas space, animated patterns are created based on their distributions rather than their movements.

In the subsequent sections, we describe the following technical features of the system: swarm simulation, drawing methods, camera-based interaction, sound synthesis, and choreographed transition of parameters. We conclude this paper by describing our observations during an experimental exhibition and outlining possible directions for future extension.

2. Swarm simulation

In Identity SA, visual and acoustic output is created via a swarm simulation. A swarm consists of a collection of mobile agents each of which behaves according to simple rules. These rules take only the interaction between neighboring agents and the local environment into account. Swarm simulations are mainly used as a computational model for group movement behaviors observed in nature such as for example in a school of fish, a flock of birds, a herd of herbivores, a colony of social insects, or a crowd of people. Simultaneously, it is successfully applied as an optimization technique for distributed systems such as in transportation scheduling and communication network routing [7]. Similar to other models of complex systems, swarm simulations can be employed to create surprising results despite the fact that they are based on entirely deterministic algorithms. For these reasons, swarm simulations can be of interest for generative art.

Idendity SA implements agent behaviors that are based on the classical BOID's algorithm [8]. Accordingly, each agent is controlled by a set of forces that causes the following behaviors:

1. collision avoidance between agents
2. velocity alignment with neighboring agents
3. flock cohesion by attraction towards the center of the neighboring agents
4. collision avoidance with the boundaries of the agent world.

The repelling forces for collision avoidance are proportional to the inverse of the square of the distance between the agent and the obstacle. Based on the sum of all the forces that affect an agent, a goal angle and a goal speed are calculated. The agent tries to meet these goal values by modifying its current orientation and velocity within the allowed limitations of its steering angle and acceleration.

Similar to the implementation in our previous project "Flocking Orchestra", the agents are divided into two species. This approach leads to a richer variety in behaviors as compared to classical BOID's type of swarms. Each species has its own individual set of parameters that control both behavior and appearance. A single parameter globally controls the interaction among agents of the same and different species. This parameter ranges from −1 to 1. A negative parameter value causes agents of one species to treat agents of the other species as moving obstacles. In this situation, no alignment or cohesion behavior among agents of different species takes place. If the parameter value is positive, even agents of the same species treat each other as obstacles. A parameter value of zero causes agents to engage in normal BOID's type of behavior that ignores species differences. This single parameter therefore allows us to control the amount of mixing or separation among agents of the two species.

The swarm consists of more than 2,000 agents and covers the whole area of the screen surface. Agents are moving in a two dimensional Euclidean space that is bounded by the rectangular borders of the screen. Due to recent improvements in computational power, it became feasible

to simulate such a large swarm in real time on a single personal computer. The system is currently implemented on an Intel-based 2 GHz Core 2 Duo CPU running MacOS X 10.4.

To calculate forces among agents, neighborhood relationships need to be determined. If we used an exhaustive algorithm to check the distance between every possible pair of agents, the computational time complexity would be proportional to square of the number of agents. To reduce this complexity, we introduced a method that divides the 2D space into a number of sub-areas. Each of these sub-areas maintains information about the agents contained within. For this reason, we can restrict distance calculations to agent pairs that reside within neighboring sub-areas.

3. Drawing methods

The agents' positions and movements are transformed into a "painting" by employing a variety of different visualization techniques and styles. The agents themselves can be drawn in one of the following five different styles.

1. a triangle pointing toward the agent's movement direction
2. a textured surface depicting a predefined pattern
3. spray spots that are randomly distributed around the agent's position

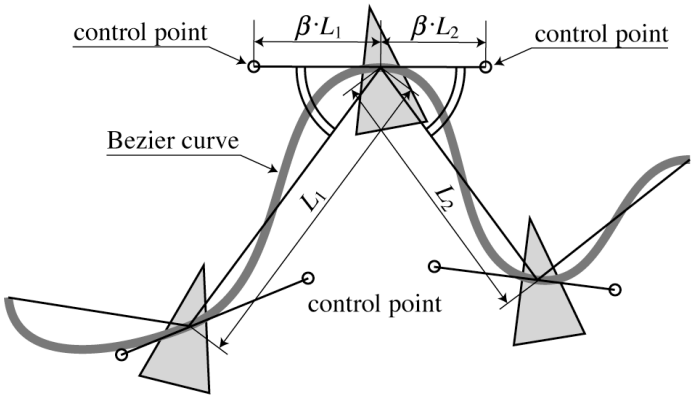
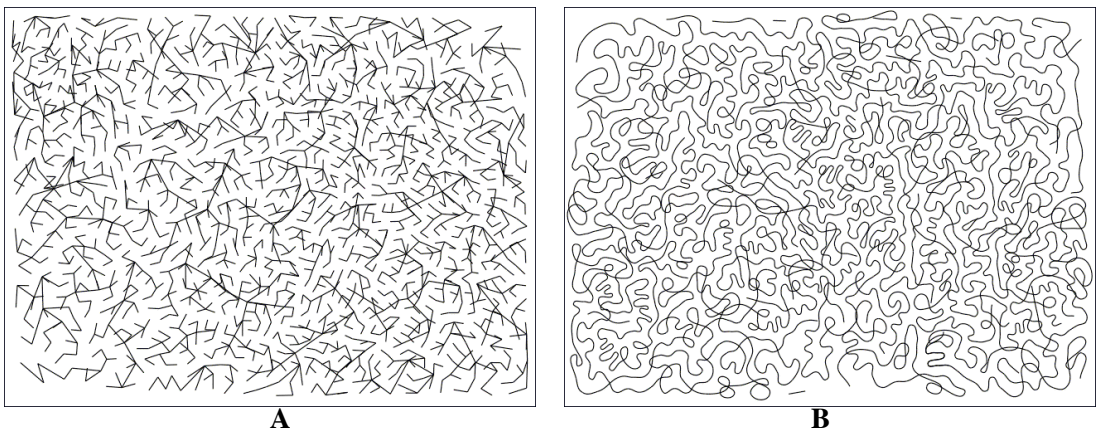


Figure 1. Illustration of Bezier curves to connect the positions of the agents smoothly.



A **B**

Figure 2. Examples of drawing line segments that connect the positions of agents. Figure A depicts connections between nearest neighbors as straight line-segments. In figure B, these connections are depicted by curved lines. Line segments above a certain length threshold are not displayed.

4. short line segments that are parallel to the agent's movement direction
5. a glyph that is selected from a predefined set of characters of Roman, Japanese and Chinese characters.

Each of the shapes listed above is drawn in a single color that is selected based on the agent's orientation angle. This mapping from angle to color depends on a parameter that can be changed via a scripting language (see section 6).

The connections between agent positions can be displayed in two different styles:

6. a set of straight lines that connect each agent to its nearest neighbor
7. a single curved line that forms a chain through all agents' positions.

The curved line is a sequence of cubic Bezier curves that smoothly connects the positions of agents. Figure 1 illustrates how the curved line is organized.

Because the computational time complexity to find the shortest path to chain the agents is very high, we employ an alternative algorithm for faster calculation. The algorithm theoretically needs a computational time that is proportional to an exponent of the number of agents in order to find the true shortest path. In the first step, it picks an agent as the start point of the line, finds the nearest agent, and connects the positions of these two agents by a line segment. Then it iterates the same calculation by finding the next position from the remaining agents. The algorithm continues to extend the line until all of the agents are chained. Fortunately, it is possible to reduce the time complexity of this algorithm by exploiting the benefits of our sub-area space partitioning structure to quickly find neighboring agents. Figure 2 depicts examples of these two types of drawing styles. Each line segment is drawn in a color that is determined by the orientation of the agent at the start point of the line segment.

Furthermore, post-processing effects such as blur and afterimage are also applicable. Combinations of these effects and different transparency settings allow the creation of a wide variety of aesthetically complex images.

Calculation speed is dependent not only on the number of agents but also on the chosen graphic styles and the display resolution. The 2 GHz machine is fast enough to produce frames at VGA (640 by 480 pixels) resolution. A 3 GHz quad-core machine with a high performance graphics processor unit is needed to produce visual output for high definition video displays which are increasingly becoming popular.

4. Camera-based interaction

Camera-based interaction is based on a simple motion detection algorithm that affects the agents' movements as well as their coloring. Basically, motion is detected by calculating the color difference between the previous and current frame for each pixel. To avoid errors in motion detection due to noise and light fluctuations, frames are cumulated into an image memory buffer by calculating a weighted sum between the memory buffer and a newly acquired frame. Expression (1) denotes the modification of each color component value M for each pixel in the memory buffer by combining it with the corresponding component I in the input frame.

$$M \leftarrow \gamma \cdot I + (1 - \gamma) \cdot M \tag{1}$$

γ is a constant coefficient in the range (0, 1). The current implementation uses a γ value of 0.1.

The next step in motion detection involves the calculation of the absolute scalar value of the pixel difference between subsequent frames. There exist several alternative methods to obtain a scalar value from the three color components (red, green, and blue) of a pixel. The current implementation selects the maximum value among the absolute differences of each color component as shown in the following equation.

$$m = \max\left\{\left|I_R - M_R\right|, \left|I_G - M_G\right|, \left|I_B - M_B\right|\right\} \quad (2)$$

Thus, this algorithm results in the creation of a single channel image that encodes the amount of detected motion in each pixel. Each agent responds to motion by adding a force towards the center of detected motion that lies within its local neighborhood.

In the absence of interaction, an agent's color is solely determined by its orientation as described in the previous section 3, but whenever the tracking system detects a visitor's motion, the agent's color is additionally affected by the corresponding pixel color in the camera frame. The ratio of orientation and interaction based color change is proportional to the amount of detected motion. Therefore, the visitor's figure is reflected when he/she is moving, but it disappears when he/she stops moving.

Figure 3 depicts the example images of analysis and synthesis. The motion distribution at the top right of the figure is calculated from camera image at the top left, and the motion distribution affects the agents' distribution show at the bottom right. Then the final image at the bottom left is composed from the agent's distribution and the original camera image.

The computational cost of motion detection depends on the resolution of the camera image. We use two different camera frame resolutions for color selection and motion detection. Colors are obtained from a camera frame in QVGA (320 by 240 pixels) resolution. A smaller camera frame of 160 by 120 pixels serves as bases for detecting the visitor's motion. This lower resolution has proven to be sufficient for our purposes in previous experiments.

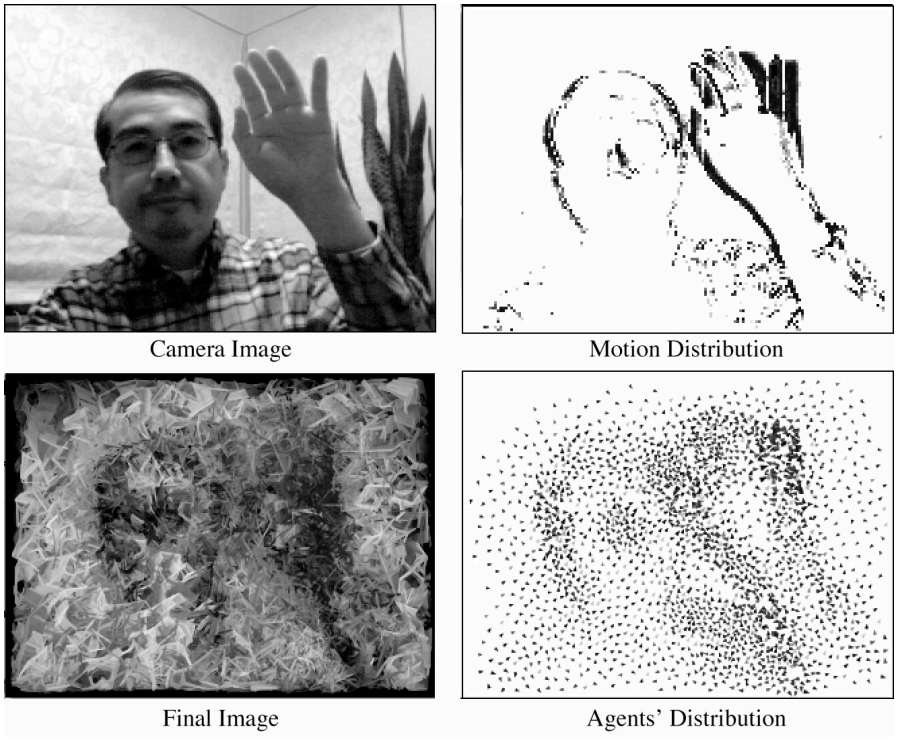


Figure 3. Camera-based interaction.

5. Sound synthesis

Sound synthesis also depends on the movement of agents and visitors. Only a subset of all agents is involved in the generation of sound. The probability that an agent creates a sound is proportional to the square of its angular velocity. Sound synthesis is controlled by several parameters such as frequency, amount of frequency fluctuation, mixing ratio of harmonic overtones and so on. All these parameters are controlled by the state of the agent. In addition, white noise is added to the overall audio output. The amount of white noise is proportional to the amount of visitor's motion.

Our current sound synthesis implementation doesn't rely on any synthesis libraries but is based on relatively simple routines that calculate samples at a fixed frame rate of 44.1 kHz. Agents that are allowed to generate sounds are organized in a queue of fixed length. In the current implementation, the queue can hold 12 agents. If a new agent is selected for sound synthesis and the queue is already full, the new agent replaces the agent that has been in the queue for the longest time.

The value of each sound sample s_t at time t is calculated according to the following equation:

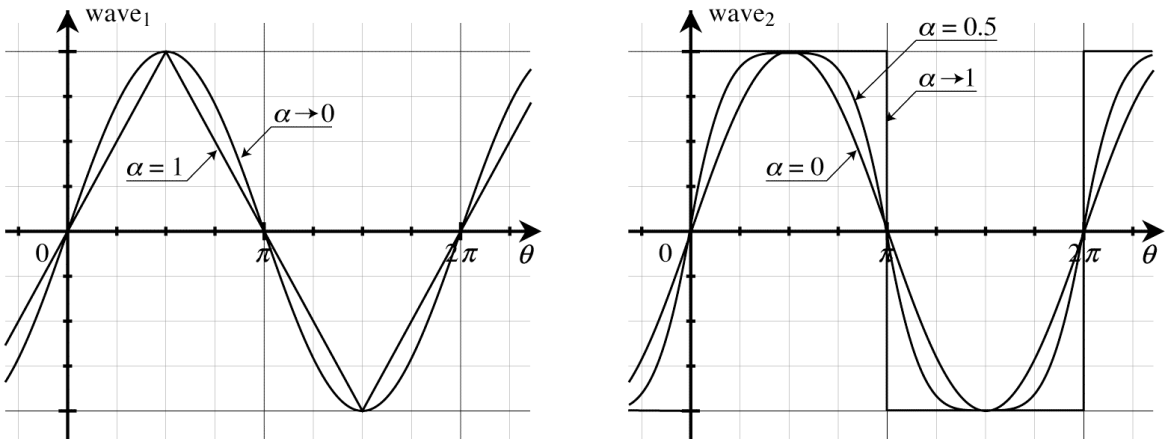


Figure 4. Functions for waveforms of sound generation.

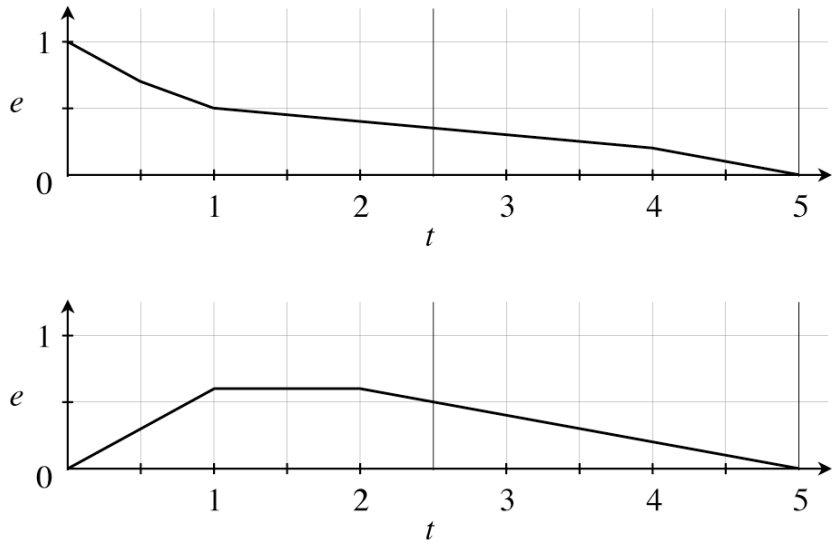


Figure 5. Shapes of two predefined envelopes.
The upper one possesses a strong attack, and the lower one starts slowly.

$$s_t = \text{wave}(a_t + A \cdot \sin b_t) \cdot (1 - B \cdot (1 + \sin c_t)) \cdot e_t \quad (3)$$

where $\text{wave}(\theta_t)$ is a periodically oscillating function that accepts a phase angle as single input parameter and produces the main waveform. A and B are constants. a_t, b_t, and c_t are angles rotating with frequencies f_a, f_b and f_c. e_t is a coefficient expressing the envelope. f_a is the frequency of main waveform, f_b is the frequency of phase fluctuation, and f_c is the frequency of volume oscillation. The values of these frequencies are determined from the states of the agent. $\log f_a$ is proportional to the vertical position y, $\log f_b$ is proportional to the absolute value of angular velocity, and $\log f_c$ is proportional to the velocity. The ranges are $40 < f_a < 1000$ (Hz), $2 < f_b < 10$ (Hz), and $100 < f_c < 1000$ (Hz). The constant A is 0.2π, and B is 0.2 in the current program.

We have created two types of "wave" functions and two types of envelopes from which each agent selects its own function and envelope. The variations of the "wave" function are:

$$\text{wave}_1(\theta) = \frac{\text{asin}(\alpha \cdot \sin\theta)}{\text{asin}\,\alpha} \quad (4)$$

$$\text{wave}_2(\theta) = \text{sign}(\sin\theta) \cdot \left(1 - (1 - |\sin\theta|)^{1/(1-\alpha)}\right) \quad (5)$$

where α is a parameter that varies from 0 to 1 and whose value is proportional to the square of velocity of the agent. Figure 4 depicts these functions for different value of α. Intuitively the generated sound becomes sharper with increasing α since the number of higher harmonic overtones increases.

The first envelope possesses a strong and fast attack like for example the plucking of a guitar string, The second envelope has a slowly increasing attack section the resembles for example the bowed sound of a violin. Figure 5 depicts the shapes of these two envelopes.

In each case, the length of the sound is scaled so that it is inversely proportional to $1 - F$ within the range of 0 to 0.9, where F represents an agent's the force of attraction towards detected motion. Accordingly, strongly attracted agents tend to produce short sounds whereas agents that don't respond to interaction create long sounds.

Finally, white noise is mixed together with the audible output. The volume of the noise is proportional to the amount of motion captured by the camera. The stereo panning of the noise is controlled by the position of the center of gravity of the detected motion.

6. Choreographed transition of parameters

The swarm simulation, its visualization and sound generation can be controlled via scriptable parameter changes. Scripting therefore offers an easy approach to create choreographed transitions in the swarm's visual and acoustic appearance. The implementation of the scripting functionality is fairly easy since our software's implementation is based on the Cocoa framework of MacOS X. The scripting functionality of our software is based on program modules for inter-process communication via AppleScript. AppleScript is an easy scripting language widely used in MacOS. For example, in order to change the agents' drawing method to triangles, the single line "set drawing style to triangles" needs to be inserted into the script. The number of scriptable parameters includes eleven parameters for visualization, eight parameters for behavior, and eight parameters for sounds. Both parameter changes and their timing can be freely chosen and therefore allow to create a wide variety of choreographed scenarios. Figure 6 is an example of the script code that iterates alternation of the shape of the agents and behavioral parameters.

A particularly interesting trasition is achieved by changing the parameter that controls the interaction among the two agent species. Depending in this parameter setting, agents from different species tend to mix uniformly or separate into distinct clusters. Figure 7 depicts an example of this visual transition.

```
tell application "DT4"
  repeat
    set species to species1
    set drawing style to curves
    set cohesion to 40
    set species to species2
    set drawing style to textures
    set avoidance to 20
    delay 10
    set species to species1
    set drawing style to bristles
    set cohesion to 50
    set species to species2
    set drawing style to branches
    set avoidance to 50
    delay 15
  end repeat
end tell
```

Figure 6. Example of scripting code in AppleScript for parameter changes.

7. Concluding remarks

We organized an experimental exhibition at Soka University during the campus festival on the 7th and 8th of October 2007. Figure 8 shows photographs taken from the exhibition space. Our software was running on a MacPro Quad Xeon 3 GHz to which a 32 inches LCD display, an iSight camera, and a pair of active speakers was connected. This computer's computation power is sufficient to simulate 2,000 agents at a rate of 25 frames per second and a resolution of 1024 by 768 pixels. Our observations of the visitor's reactions have been very encouraging. New visitors quickly understood the interaction possibilities of the system when they perceived how their reflected image emerged in the visual output. Some children and young persons then started to move their body more actively or even began to dance in front of the screen. Typical visitor comments were: "it is interesting" and "I never tire of this." On the other hand, one visitor wasn't particularly happy about the acoustic feedback and said: "I feel getting sick from hearing such a sound". From am artistic viewpoint, this comment is not that bad because it is evidence that the work made a strong impression on the visitor.

We would like to further extend the system. First of all, we would like to experiment with alternative mechanisms for generation of sound such as playing back prerecorded and life sampled sounds. Furthermore, we would like to setup the installation in non-exhibition

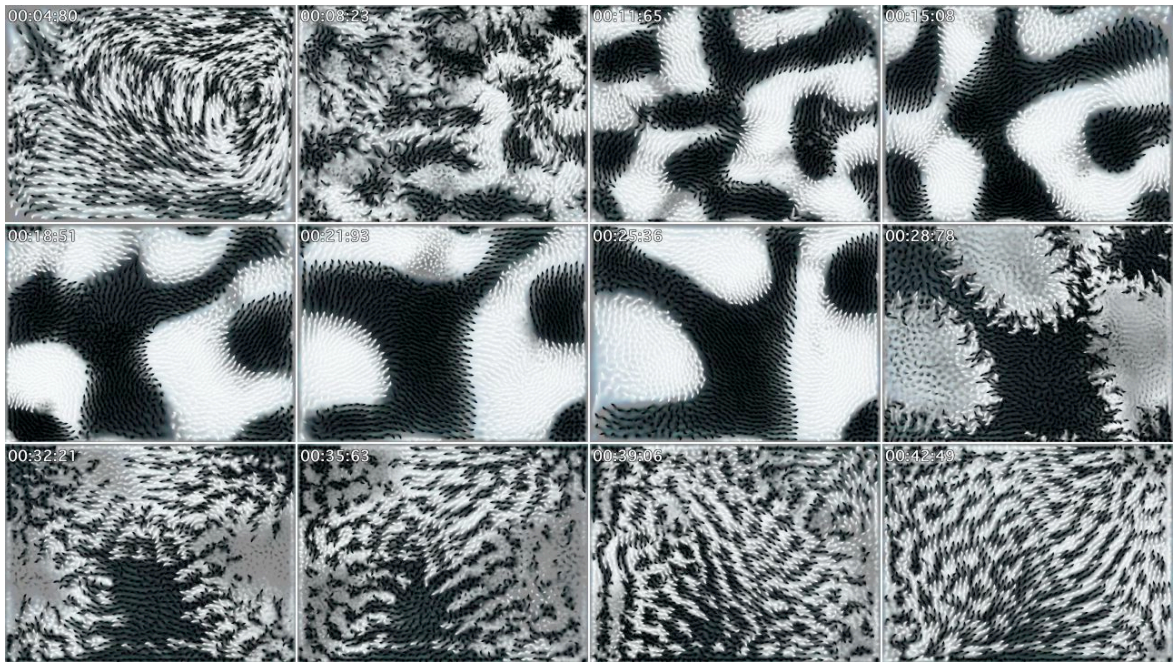


Figure 7. Example of visual transition by changing the parameter value of inter-species interaction.

Figure 8. Photographs taken from the exhibition room in the campus festival of Soka University on 8th October 2007.

environments such as in public space or on stage where it could be combined with a dance performance.

To conclude, we hope that Identity SA creates an aesthetic experience that provokes visitors to reflect on issues of identity and transitoriness. The binary executable and sample scripts are downloadable from the link in the project web site:

```
http://www.intlab.soka.ac.jp/~unemi/1/DT4/
```

Acknowledgement

The authors would like to thank students in Soka University and all of visitors who supported and participated the experimental exhibition in the campus festival, and Prof. Dr. Rolf Pfeifer who is supporting this international collaboration. This work is partially funded by Grants-in-aid for Scientific Research #17500152 from Ministry of Education, Culture, Sports, Science and Technology (MEXT) in Japan.

References

[1] Daisaku Ikeda, The Living Buddha – an Interpretive Biography, trans. Burton Watson, Weatherhill, New York, p. 19, 1976.

[2] Tatsuo Unemi and Daniel Bisig, Playing Music by Conducting BOID Agents - A Style of Interaction in the Life with A-Life, in J. Pollack et al eds. Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, MIT Press, Boston, MA, pp. 546-550, 2004.

[3] Tatsuo Unemi and Daniel Bisig, Flocking Orchestra – to play a type of generative music by interaction between human and flocking agents. in C. Soddu ed. Proceedings of the eighth Generative Art Conference, pp. 19-21, 2005.

[4] Daniel Bisig and Tatsuo Unemi, MediaFlies – A Video and Audio Remixing Multi Agent System. in C. Soddu ed. Proceedings of the 9th Generative Art Conference, pp. 63-74, 2006.

[5] Tatsuo Unemi and Daniel Bisig, Flocking Messengers, –, pp. 272-280, 2006.

[6] Christian Jacob, Gerald Hushlak, Jeffrey Boyd et al, SwarmArt: Interactive Art from Swarm Intelligence, *Leonardo*, Vol. 40, Issue 3, pp. 248-255, 2007.

[7] Gerardo Beni, From Swarm Intelligence to Swarm Robotics, in *Swarm Robotics*, Springer, LNCS 3342, pp. 1-9, 2005.

[8] Craig W. Reynolds, Flocks, herds, and schools: A distributed behavioral model, *Computer Graphics*, Vol. 21, No. 4, pp. 25-34, (SIGGRAPH '87 Conference Proceedings) 1987.