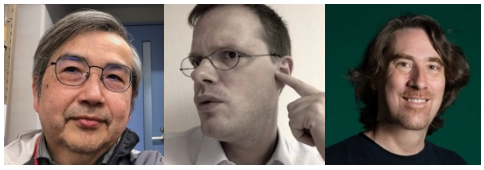
Proceeings of the 28th Generative Art Conference,

125-133, Rome, Italy, Dec. 16-18, 2025.

Phantom in the Box – an Interactive Installation with Large-Scale Flocking Agents

Prof. T. Unemi, DEng.
Glycan and Life Systems Integration Center, Soka University, Hachioji, Japan
https://unemi.vivian.jp
e-mail: unemi@me.com

Prof. P. Kocher, PhD. and Dr. D. Bisig, PhD.
ICST, Zurich University of the Arts, Switzerland



Abstract

Phantom in the Box is an interactive audiovisual installation that invites visitors to explore the transient and delicate organization of their own physical form. It provides an artistic exploration of human existence as a spatiotemporal process interwoven with the surrounding environment. Human beings exist as unique and long-lived entities, yet their physical composition is in constant flux, with matter continuously entering and leaving.

The system displays a dynamic swarm of colorful particles within a white cube. In response to the visitor's presence and movements, the particles disperse into a diffuse cloud or coalesce into a representation of the visitor's body.

The system utilizes a large-scale BOIDS model to simulate swarm behavior. The BOIDS model is run on a GPU to achieve real-time performance for up to one million agents. A depth camera tracks the visitor and generates a point cloud representing his/her body surface. These points are integrated into the BOIDS model, allowing the simulated agents to react dynamically to the visitor.

The audio output is created by using a particle-based synthesis technique. To control the generation of audio, statistical features of the simulated agents are applied to the synthesis parameters.

1. Introduction

Simulation techniques to model the collective behavior of bird flocks have been widely adopted in computer-based new media art to create dynamic and complex patterns.

Recent advances in computer technology have enabled compact personal computers to simulate millions of autonomous agents. These large-scale simulations produce emergent formations that qualitatively alter human perception of patterns, momentarily drawing attention to macro-level structures before revealing that these structures consist of individual entities. This perceptual shift aligns with the psychological phenomenon known as the *Global Precedence Effect*, first identi-

fied by David Navon [1].

Each living individual exhibits a paradoxical existence characterized both by permanence and impermanence. The individual is a unique and irreplaceable entity, but its body is constituted by transient chemical compounds that assemble temporarily within organs and eventually depart as excretions. The continuity of the individual emerges only as a spatio-temporal process in dynamic relation to the surrounding environment and other entities.

Our previous work, *Identity-SA* [2, 3], provided visitors with the experience of observing a fragile organization of bodies reflected within a two-dimensional mirror. This visual and sonic output of the work was generated by a simulated swarm that moves across a planar surface. The swarm responded to the visitors' presence, which was analyzed using real-time image-processing algorithms.

The new work, *Phantom in the Box*, provides visitors with a similar setting but extends it into three-dimensional space. The simulated swarm and the visuals it generates are situated within a volumetric environment rather than on a flat surface. This environment conveys a distinctly different impression, emphasizing how the corporeal presence of the visitor becomes malleable through the swarm's reactions to the visitor's gestures.

As an installation, *Phantom in the Box* runs on a combination of two (or three) small computers, a display (a large screen or a video projection), and a depth camera. A large swarm consisting of several hundred thousand agents is simulated on the main computer and controls the 3D visuals. This computer also calculates several statistical measures that describe the movements of the agents. These measures are sent to another computer that generates sound effects. The depth camera detects the 3D shape of the visitor's surface as a 2D map representing the distance values between the camera and the visitor's body. The computer receiving this map computes a gradient map from which it generates a point cloud consisting of several thousand points that uniformly cover the detected surface within 3D space. Information about the positions of these points, along with corresponding color values obtained from the RGB camera attached to the depth camera, is transmitted to the main computer so that the swarm can react to the visitor's gestures. Figure 1 shows a screenshot of the visual rendering in which the visitor's shape appears faintly within the colorful swarm. Figure 2 illustrates a possible installation setup that employs an ultra-short-focus projector placed on a table.

Figure 1. Screenshot of the installation's visual output.

Figure 2. Illustration of a possible exhibition setup of the installation.

The following sections describe the model and simulation of a large-scale flock, the interaction mechanism, and the design of visualization and sonification.

2. Simulation of a Flock

Flocks of birds, schools of fish, and herds of herbivores are well-known natural phenomena that can be observed in large groups of individuals that interact with each other. To understand the underlying mechanisms causing these phenomena, extensive research has been conducted across various fields such as biology, robotics, computer science, and mathematics. This research has led to the development of mathematical models and efficient algorithms to simulate these phenomena. Some of the simulations have been adopted for creating computer animations and computer games. In 1987, Craig Reynolds proposed the pioneering BOIDS model [4] and applied it to create a 3D computer graphics short film entitled *Stanley and Stella in: Breaking the Ice*.

The flocking simulation employed in *Phantom in the Box* largely follows the original BOIDS model. Each agent i possesses a position p_i and velocity v_i. At each simulation step, a force vector F_i is applied to the agent. This vector is calculated as a combination of three factors, separation, cohesion, and alignment, which are obtained by observing other agents that are close and within the field of view. The separation factor prevents collisions between individuals. The cohesion factor causes agents to move toward the neighboring agents' center and maintain a compact structure. The alignment factor causes agents to adjust their direction of motion to match that of their neighbors. The resulting force F_i is calculated as a weighted sum of the three individual factors.

$$F_i = W_{sep} \sum_{j \in A} \frac{p_i - p_j}{D_{ij}^3} + W_{coh} \left(\frac{\sum_{j \in A} p_j / D_{ij}}{\sum_{j \in A} 1/D_{ij}} - p_i \right) + W_{ali} \frac{\sum_{j \in A} v_j / D_{ij}}{\sum_{j \in A} 1/D_{ij}} \qquad (1)$$

Here, A refers to a set of agents inside the field of view volume, and D_{ij} refers to the Euclidean distance between the agent positions p_i and p_j. The separation factor follows the formulation of the original BOIDS system, in which the repulsive force between two individuals is inversely proportional to the square of their distance. The other two factors, however, are modified so that their effects are inversely proportional to the distance. This modification helps maintain a uniform flock density within each group, whereas the original definition tends to produce higher density in the central region.

In addition to the original BOIDS model, an attraction force is introduced that

enables agents to respond to the presence and movement of a visitor. This force treats the points in the point cloud generated by the camera tracking software as additional agents. These additional agents exert a cohesion force on the flocking agents. Further details are provided in Section 3.

Each agent's position and velocity is calculated from the total force vector using simple numerical approximation. The approximation employs a revised version of the Euler method, which follows the differential equation of Newtonian mechanics. The position p'_i and the velocity v'_i of the next simulation step are calculated from the current values p_i and v_i using the following equations.

$$v'_i = \left(v_i + \frac{F_i}{m} \Delta t \right) \cdot (1 - \gamma)^{\Delta t}$$

$$p'_i = p_i + (v_i + v'_i)/2 \cdot \Delta t$$

Here, m represents the agent's mass and γ represents the friction coefficient.

The computational cost for running the BOIDS model is primarily determined by the number of calculations required to evaluate interactions between agents. Partitioning the Euclidean simulation space is effective for excluding negligible interactions based on distance. The software implementation of this system employs an efficient algorithm that integrates such a partitioning method with parallel processing on both the multi-core CPU and GPU. Further details will be presented in a subsequent paper [5].

3. Interaction

The ability to run simulations of natural phenomena in real time is essential for employing these simulations in interactive art installations or computer games. As mentioned previously, *Phantom in the Box* uses a depth camera (Intel® RealSense™ D435) to detect the 3D surface of a visitor's body. It captures a 640 × 480 2D distance map along with a color image of the same resolution, at 30 frames per second. A computer receiving this raw data periodically processes it to generate a point cloud in 3D space.

The point cloud contains several thousand points, each consisting of a 3D position and RGB color. To distribute these points across the 3D surface as uniformly as possible, a processing is employed that removes the background by applying a threshold distance, then transforms the depth map into a gradient map by calculating the absolute value of the differentiation for each 3 × 3 window. This gradient map is used to create a probability list that acts like a roulette wheel to determine the random position of each point. The algorithm randomly selects a pixel position based on the probability list. Subsequently, it places a point at a random location within the selected pixel to determine its xy coordinates. Finally, the z coordinate is calculated by linearly interpolating among the depth values of the four nearest pixels.

The points that have been determined in this manner act as additional agents that attract the simulated flocking agents. To calculate this attraction, another iteration is added to the equation (1) to test related points.

Algorithm 1 Organizing the force field.

▷ *Prepares the indexes array once at the beginning.* ◁
$i \leftarrow 1$.
for $x = 1 - m_x, \ldots, m_x - 1$ **do**
  for $y = 1 - m_y, \ldots, m_y - 1$ **do**
    for $z = 1 - m_z, \ldots, m_z - 1$ **do**
      $\mathbf{K}_i \leftarrow (x, y, z)$, $i \leftarrow i + 1$.
▷ *to randomize the elements order of the same* $|\mathbf{K}_i|$ ◁
randomizes the order of elements in $\mathbf{K}$.
sorts $\mathbf{K}$ by $|\mathbf{K}_i|$ in ascending order.

Require: $\mathbf{C}$: an array of N_g elements each of which represents a set of points in the corresponding partition.
Ensure: $\mathcal{F}$: an array of $m_x \times m_y \times m_z$ vectors that represents a force field toward the nearest partition including a point.

function SCALAR INDEX((i_x, i_y, i_z))
  return $(i_z \cdot m_y + i_y) \cdot m_x + i_x$.
function INDEX VECTOR(i)
  return $(i \mod m_x, (i/m_x) \mod m_y, (i/m_x/m_y) \mod m_z)$.
$\kappa \leftarrow 1$.
for $i = 1, \ldots, m_x \times m_y \times m_z$ **do**
  if $\exists j \in \{1, \ldots, C\}$. $\mathbf{C}_{C(i-1)+j}$ includes a point **then**
    $\mathbf{F}_i = i$.
  else $\mathbf{F}_i = -1$, $\mathbf{G}_\kappa \leftarrow i$, $\kappa \leftarrow \kappa + 1$.
for $i = 1, \ldots, (2m_x - 1) \times (2m_y - 1) \times (2m_z - 1)$ **do**
  $\rho \leftarrow 1$.
  for $j = 1, \ldots, \kappa$ **do**
    $v \leftarrow$ INDEX VECTOR($\mathbf{G}_j + \mathbf{K}_i$).
    if $\exists * \in \{x, y, z\}$. $v_* \le 0$ or $v_* > m_*$ **then**
      continue.
    $s \leftarrow$ SCALAR INDEX(v).
    if $\mathbf{F}_s \ge 0$ **then** $\mathbf{F}_j \leftarrow \mathbf{F}_s$, $\mathbf{E}_\rho \leftarrow j$, $\rho \leftarrow \rho + 1$.
  if $\rho > 1$ **then**
    $\tau \leftarrow 1$, $\sigma \leftarrow 1$,
    for $j = 1, \ldots, \kappa$ **do**
      if $\sigma < \rho$ and $\mathbf{G}_j = \mathbf{E}_\sigma$ **then** $\sigma \leftarrow \sigma + 1$,
      else $\mathbf{G}_\tau \leftarrow \mathbf{G}_j$, $\tau \leftarrow \tau + 1$.
    $\kappa \leftarrow \kappa - \rho - 1$.
  if $\kappa \le 1$ **then** break.
for $i = 1, \ldots, m_x \times m_y \times m_z$ **do**
  $\mathcal{F}_i \leftarrow$ unit vector of (INDEX VECTOT($\mathbf{F}_i$) − INDEX VECTOT(i)).

To attract distant agents to the 3D surface, a force field directed toward the nearest surface point is organized into an array of 3D vectors corresponding to the m_x × m_y × m_z partitions. The actual value used in our implementation is 16 × 9 × 12. The force field affects an agent even when no surface points are present within the agent's view volume. The force vector applied to a distant agent is calculated by interpolating among the nearest eight partition centers, using weights proportional to the inverse of the distance between each partition center and the agent's position.

Algorithm 1 combines two procedures to organize the force field. The first procedure prepares a sorted array of 3D integer vectors, where each vector represents an index offset. This procedure is executed once before the first simulation step. The offsets are arranged in order of increasing Euclidean distance from the origin. Several offsets possess the same value of the sort keys, as each offset is a permutation of three integers. For example, all eight offsets (1,1,1), (1,1,-1), (1,-1,1), ... share the same key value, namely $\sqrt{3}$. The order of offsets with the same key is randomized to prevent an unnatural regularity in the swarm's motion. The second procedure is executed each time a new set of point clouds is calculated. The vector in a partition is set to zero if that partition contains a point; otherwise, it is a unit vector pointing to the position of the nearest partition that contains a point.

A high-resolution point cloud is useful for achieving high precision, but it consumes considerable computation time and memory space. In our implementation, we use five thousand points to construct the point cloud. The positions and colors of these points are transmitted to the main computer running the simulation. This number of points ensures acceptable computational performance, memory usage, and communication bandwidth. Each point is represented by 120 bits of data, including three 32-bit floating-point numbers for position and three 8-bit un-

signed integers for the RGB color components. Therefore, a data set of (120/8) × 5K = 75K bytes is generated per frame, which requires 75K × 30 = 2.25M bytes per second to be transmitted to the simulation. A commonly used interface for wired communication between two computers, such as Gigabit Ethernet, provides sufficient capacity to handle data transfer at this rate.

To reduce the computational load on the machine running the simulation, the points in the point cloud are assigned to the same grid of partitions used in the simulation. The data for transmitting the point cloud information is grouped as a packet for each partition.

4. Visuals

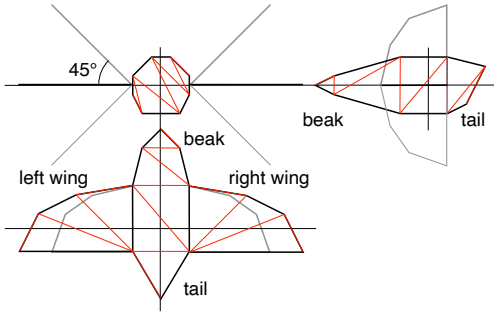


Figure 3. Graphical bird shape.

Each agent is rendered as a small graphical shape. To minimize computational cost, it is represented as a paper plane composed of two triangles: one horizontal and one vertical. An alternative bird shape is also implemented, consisting of three polygons for the body and two polygons for the wings, which are animated to simulate flapping. The three polygons representing the body are not intended to depict the faces of a polyhedron, but rather the silhouette shapes perpendicular to the three axes. Although this shape requires a higher computational cost for rendering, it helps visitors recognize that the visualization represents a flock of birds rather than a cloud of particles.

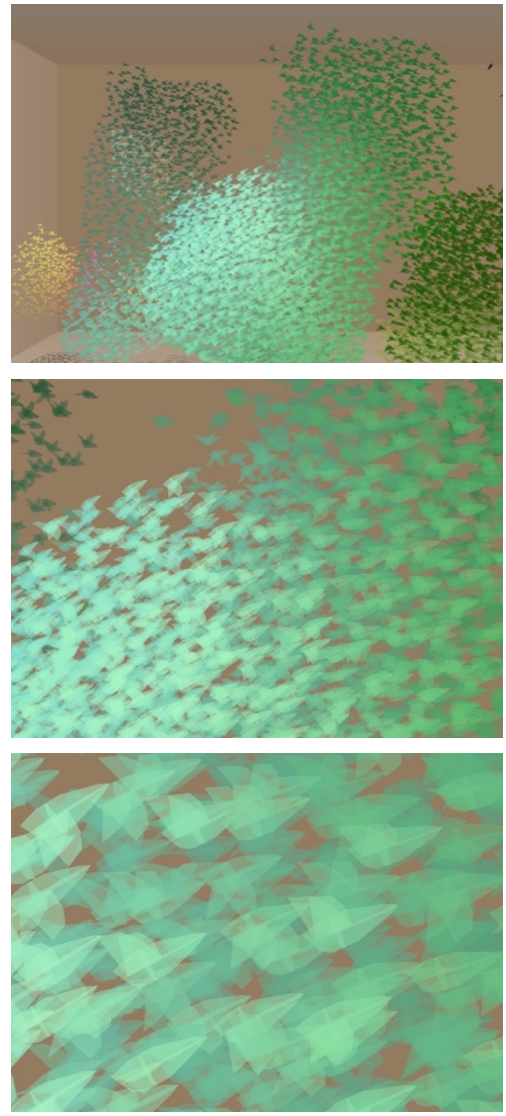


Figure 4. Screenshots of a flock of birds in different magnifications.

The flapping frequency of the bird shape alternates in response to the tilt angle of the agent. It flaps its wings more rapidly when it ascends. Figures 3 and 4 depict the shape agents that mimic a bird with wings.

The color is determined from the velocity vector by mapping the pan angle to the hue value, the tilt angle to brightness, and the speed to saturation. In other words, agents facing left are rendered in red, while those facing right are rendered in green. Brighter colors are assigned to agents flying upward, and darker tones to those flying downward. Fast-moving agents appear with vivid colors, whereas slower ones appear subdued. Due to the alignment factor in the BOIDS model, the velocity vectors of neighboring individuals tend to share similar colors. This color-mapping method produces a dynamically shifting display of vibrant colors that follows the flock's movement.

The colors of the point cloud also influence the agents when a point within an agent's view volume affects the cohesion factor. By blending the colors derived from velocity with those of the nearest point, a faint reflection of the visitor's shape becomes perceptible within the visualization.

5. Sound

For an interactive system, even when its primary form of expression is visual, the existence of a sound layer plays an important role, as it influences the perception of the system and therefore shapes the aesthetic experience of the artwork as a whole.

The visualization directly reflects the swarm's behavior and its reaction to the visitor as described above. For the sound, a metaphorical translation was needed. In the case of *Phantom in the Box,* the sound reflects the underlying BOIDS mechanism and the visual appearance that results from it, utilizing a sound synthesis technique that generates dense clouds of short sound particles. To establish a correspondence with the dynamics and interactivity of the visuals, the parameters of sound generation – loudness, timbre, and stereo position – are directly coupled to the statistical measures that describe the movements of the agents.

The statistical measures are calculated by dividing the simulation space into partitions and counting the number of agents contained within each, along with a vector representing their average velocity. For efficiency, only the x and y dimensions of the space are considered for partitioning. This results in 16 × 9 × 1 = 144 partitions. The sound generation software operates separately from the simulation. Statistical data are transmitted from the simulation to the sound generation software via OSC over a UDP/IP connection. The transmission rate is at least one data packet per second, preferably higher, to ensure smoother interaction.

The sound engine instantiates 144 synthesizers, one for each partition. Each synthesizer outputs a stream of short sound particles, generated through frequency modulation synthesis. Two parameters of each synthesizer are fixed: the range from which the grain frequencies are randomly selected, and the stereo position of the sounds. These two

parameters follow an obvious pattern: the left-right position on the screen is linked to the stereo position, while the top-bottom position is linked to the pitch.

The number of agents in one partition and their average velocity control the loudness of the synthesizer. This means that if a partition contains no agents, the associated synthesizer is muted. Furthermore, the average velocity of the agents in a partition controls the duration of the sound particles as well as several parameters of the frequency modulation synthesis that have an impact on the timbre: the ratio between the carrier and modulation frequencies, the modulation index, and the cutoff frequency of a low-pass filter. The higher the average velocity, the brighter and denser the timbre becomes.

6. Conclusion

Thanks to the performance improvement of parallel processing in small computers, it has become possible to smoothly animate a swarm of one million agents in real-time using an efficient algorithm based on the BOIDS model.

The interaction method presented here represents one possible application in which real-time responsiveness is essential. The interaction between the human body and a swarm within a virtual 3D space can effectively evoke in visitors a sense of the transience of physical existence.

Self-awareness is one of the key psychological functions central to understanding what it means to be human. It also serves as a foundation for reflecting on individuality, social responsibility, and the design of political systems. Yet, we often misinterpret the spatiotemporal continuity and extension of individual existence. In recent years, not only fictional narratives in science fiction but also the physical realization of intelligent humanoid robots have confronted us with such complex and controversial questions.

We believe that one of the vital roles of artistic creation related to artificial life research is to provide unique experiences that deepen human reflection on life itself. We hope that as many visitors as possible will not only enjoy this artwork but also find inspiration for new challenges in both scientific research and artistic creation.

References

[1] Navon, D., Forest before trees: The precedence of global features in visual perception, *Cognitive Psychology*, Vol. 9, No. 3, 353–383, 1997. https://doi.org/10.1016/0010-0285(77)90012-3

[2] Unemi, T. and Bisig, D., Identity-SA – an interactive swarm-based animation with a deformed reflection. In Soddu, C., editor, Proceedings of the Tenth Generative Art Conference, 269–279, Milan, Italy, 2007.

[3] Unemi, T., Matsui, Y., and Bisig, D., Identity SA 1.6 – an artistic software that produces a deformed audio-visual reflection based on a visually interactive swarm. In Proceedings of the ACE 2008 International Conference on Advances in Computer Entertainment Technology, 297–300, Yokohama, Japan, 2008. https://doi.org/10.1145/1501750.1501821

[4] Reynolds, C. W., Flocks, herds, and schools: A distributed behavioral model. Computer Graphics, 21(4):25–34, 1987. https://doi.org/10.1145/37402.37406

[5] Unemi, T., An Efficient Algorithm Without Data Cache to Simulate a Large Scale Flock in 3D Space, AROB 2026, Beppu, Japan, to be published.