Loops was commissioned as a "digital portrait" of choreographer Merce Cunningham, and takes as its point of departure a motion-capture recording of Cunningham performing his 1970s solo dance for hands of the same name. It is a piece for the screen but has been presented simultaneously with the parallel work Loops Score *which provides a related soundtrack.*

Chapter 3 — *Loops*

Cunningham originally created *Loops* as a solo to be performed in front of a Jasper Johns painting at the Museum of Modern Art.

"Described by Cunningham as an 'Event for soloist,' *Loops* was performed by him at the Museum of Modern Art, New York on 3 December 1971... The piece was performed in front of Jasper Johns's large painting *Map, after Buckminster Fuller's Dymaxian Airocean World*, in the Founders' Room on the museum's sixth floor... *Loops* was performed again at New York's Whitney Museum of American Art on 18 May 1973 (as *Loops and Additions*), and it also gave Cunningham material for his appearances in Event performances, such as the solo in which his hands move through the air around his head and torso, fingers flickering and twitching... "

D.Vaughan, *Merce Cunningham: 50 Years,* Aperture, New York, 1999.

Loops is a portrait of Cunningham — it attends not to his appearance, but to his motion. It is derived from a motion-captured recording of his 1971 solo dance for hands and fingers entitled *Loops*. In this work, his motion-captured joints become nodes in a network that sets them into fluctuating relationships with one another, at times suggesting the hands underlying them, but more often depicting complex cat's-cradle variations. These nodes render themselves in a series of related styles, reminiscent of hand-drawing, but with a different sort of life. Many viewers liken their experience of seeing *Loops* to that of gazing into nature: its flickering motions put them in mind of fire or of primitive biology, perhaps seen under a microscope.

Loops is computed in real time and is, in effect, a live performance (the program is the only "performer" of this choreography other than Cunningham, who has never set the work on any other dancer.) Thus *Loops*, the digital program, confers an odd kind of immortality on *Loops*, the physical dance, for in essence it keeps improvising itself. Manifesting itself through the probabilistic interaction of its distinct parts, it reveals something new with every playback.

1. An overview of the artwork

> Art is the imitation of nature in her manner of operation.

This quote from **Thomas Aquinas** seems to be a favorite quote of Cage's (sometimes attributed by Cage to Coomaraswamy).

c.f. **J. Cage,** *Composition as Process, Changes* and *On Robert Rauschenberg* collected in : *Silence,* Wesleyan University Press, 1961.

c.f. **T. Aquinas,** *Summa Theologica*.

This idea, cited often by collaborators John Cage and Merce Cunningham, led them to a deeper kind of realism, one that mirrored not the world's outward appearance, but rather its underlying processes. One such process, which fascinated both artists, was the workings of chance. They decided that by leaving many of their creative decisions to the roll of the dice, they could give their artworks true autonomy.

Loops explores, questions, and then extends these radical notions of realism and autonomy. *Loops* is a work about distribution and change, about distributing networks of coherence over underlying human motion that are never stable, but the remains of which become the material used to create new networks. There is a minimalism to the monochromatic imagery, but it is accompanied by a sense of finiteness of material — that there is a limit to the number of lines and the number of points available and that the piece, which has no beginning or end, is inefficiently enumerative.

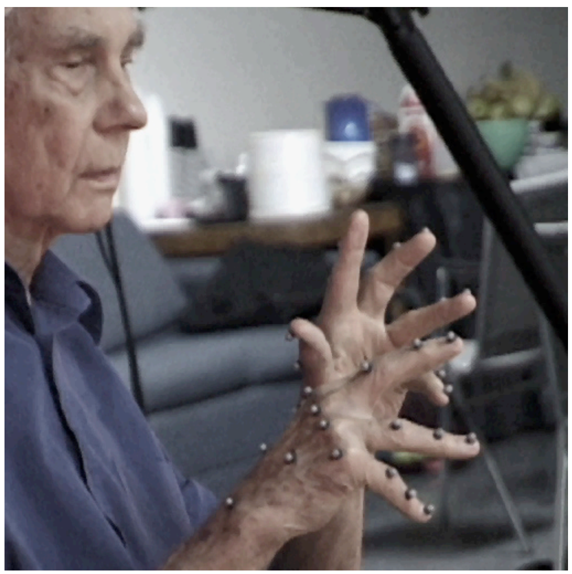


figure 20. Cunningham performing *Loops* for motion-capture session in 1999.

In its particular fashion, the work indicates the first direct point of contact between the history of chance operations in digital art, in which the long time collaborators Cunningham and Cage play a significant role, and the probabilistic action selection strategies of artificial intelligence. The agent metaphor enters this work through the motion-capture points — some 42 points are represented in the original motion-capture session, and we distribute 42 simple creatures as a "colony" across the hand data. This reflects the desire for an complexity that was entirely interior to the moving hands (this piece is *live* but not *interac-*

tive with the viewers). The desire to explore, and given the unlimited duration of this piece potentially exhaust, the making and unmaking of relationships between the finite number of points on the hands implies that the creatures' perceptual world should be quite rich — for what the creatures sense of their peers is the hidden underpinnings of these fluctuating relationships.

Motion capture is the name given to a technology that uses multiple, calibrated cameras to reconstruct the three-dimensional motion of points in space. These points of motion are typically markers attached to human movers. When "cleaned" offline, by hand, motion capture offers a sometimes astonishing accuracy of reconstruction of moving skeletons — the kind of fidelity appreciated by motion-capture's roots in both the biomedical community and military simulation. Today it typically provides the source material for the animations found in computer games, and Hollywood 'digital extras'.

Most recently commercially available real-time motion capture has become a practical reality — and systems made by three hardware manufacturers are now available.

Motion capture hardware, however, remains within the budget scale of Hollywood, the military and computer games. *Loops* used offline-motion capture of a performance by Merce Cunningham that was generously donated by Modern Uprising Studios. The two other dance works use real-time motion capture, with hardware and engineering support donated by MotionAnalysis Corporation.

Therefore, their perceptual worlds included the movement of the motion capture point it is associated with an individual creature, a number of senses of its local neighborhood of points and signals sent directly from other creatures. These latter two senses were additionally available in forms weighted by the existence of visible connection between the creature being perceived and the creature doing the perceiving. Signaling in the micro-world of the piece takes place at a finite speed, inside a simulated virtual fluid. Thus signals propagate and join in waves throughout the space of the colony and, as they push the behavioral tendencies of the creature around, these signals are also rendered visible on the creatures bodies.

Indeed the visual appearance of the work stems as much from rendering the perception of these creatures visible as it does from allowing the creatures to chose their appearance, but from the perspective of our agent framework in these creatures' bodies we find an important example of a heterogeneous, non-movement oriented motor system. Each creature's "body" consists of: a set of ordered lists of points that each start with the creature's own point (for example, these may be drawn as connected line segments), a set of filter coefficients applied to a motion sampler that samples the underlying motion capture data, and a position in a blend space of rendering styles.

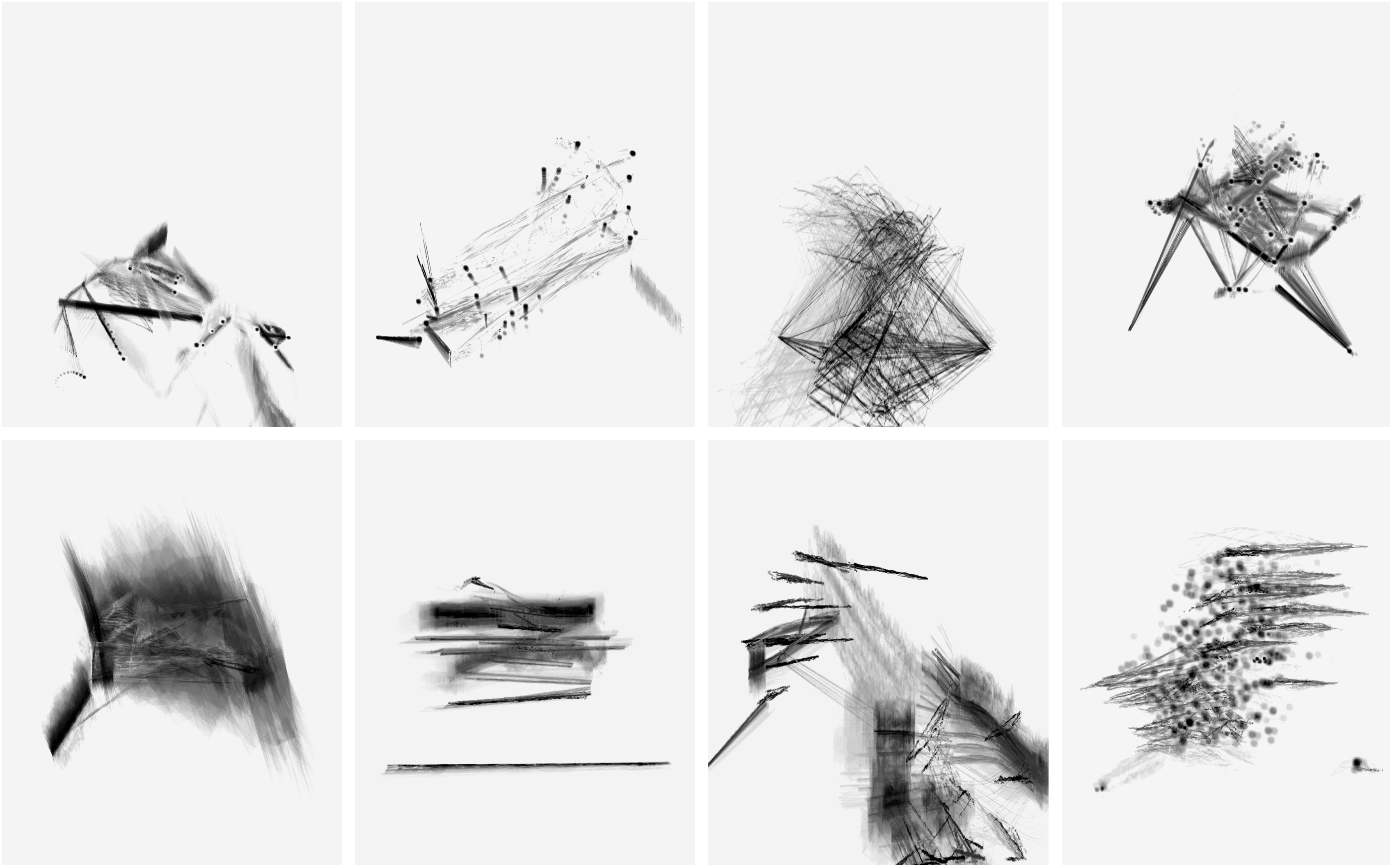


figure 21.
Loops (inverted).

Since we are concerned with the distribution of similarity and difference, much of the technical and computational resources deployed in *Loops* are concerned with the action systems of our creatures. These systems are constructed using a probabilistic action selection framework very similar to c43, *page* 55. Recall that the two basic construction units of this hierarchical behavior system is the *action-group* and the *action-tuple*.

The contents of each of these parts will be the subject of much description: the triggers for each action-tuple come from a perception of what other creatures in the colony are doing — this is a signaling mechanism global throughout the colony; the do-while is a duration distribution that starts out set by hand; similarly the values of the action tuples are also hand set initially.

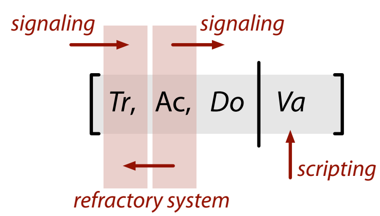


figure 22.
In *Loops* the action tuple is augmented with two structures. the trigger system and action collaborate in a refractory mechanism and a signaling process. These signals couple the behavior of the agents in Loops to each other.

Two extra elements are added to this basic configuration. First is a *refractory multiplier* for the action — actions that fire are less likely to fire again. This dodges many of the coupling / temporal pathology problems discussed in the critique of the c43/c5 action selection strategies, *page 71*, effectively damping away any chance of a one-iteration dither, which otherwise may occur since this action system is coupled through the signaling mechanism to 41 others.

The second element specific to *Loops* is an expectation mechanism that cuts across the action-tuples' duration controlling do-while — in the event of something "surprising" (to be defined below) occurring, a re-selection is almost certain to occur.

The strength of coupling between each of these two systems and the action-tuples are adaptive: the refractory systems adapt their time constants to be twice the ultimate durations of each individual action-tuple; the expectation mechanisms adapt the size of their effect (equivalently what the expectation mecha-

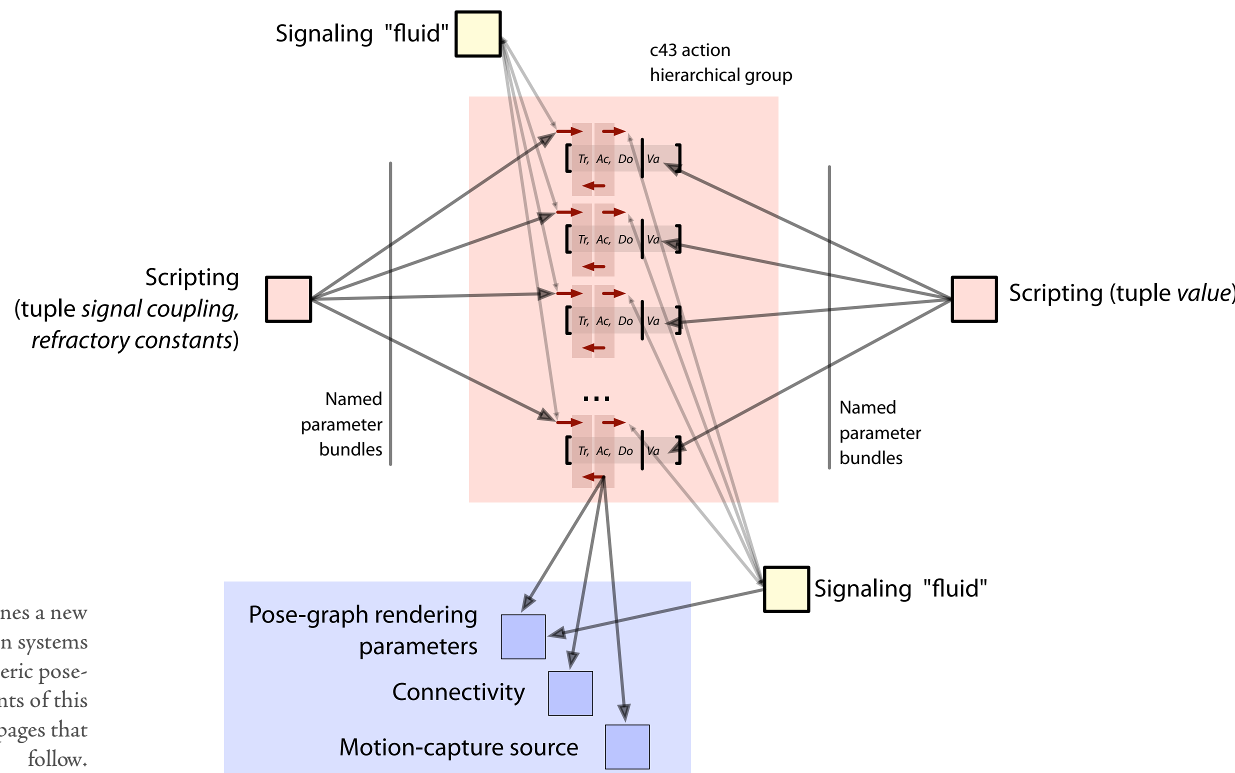


figure 23. Each *Loops* agent combines a new set of scripting techniques for action systems with a new exploitation of the generic pose-graph motor system. The contents of this diagram will be discussed in the pages that follow.

nisms consider truly surprising) to try and keep the typical duration of an action-tuple to be near an author-specified duration (10 seconds). These internal adaptive parameters had, of course, the ability to be reset externally — thus these resets had the ability to be *scored*.

The motor systems

Inside each action-tuple are, of course, the actions themselves. In *Loops* these take the form of (possibly a composite of) one of three classes — asking the creature's motor system to change **rendering style**, asking the creature's body to change its **connectivity** or asking the creature to **send signals** out into the colony. We'll look at each of these possible actions in turn.

The creature's primary motor system is constructed purely within the pose-graph framework, but unlike earlier work using these structures, exploits the representation neutrality of the pose-graph not to render the realistic animation of a representational creature using combinations of pre-made material, but rather to control the rendering parameters of a non-representational creature using combinations of pre-made sets of parameters.

Thus we fill in the lower levels of the otherwise abstract pose-graph with the following structure:

The **pose representation** — an *ad hoc* collection of rendering parameters, line styles, noise amplitudes, couplings to signals. These bundles are precursors to other persistent structures in the agent toolkit, the persisted partial trees, *page 209*, and the long-term learning database, *page 127*.

As we have seen, this pose-graph motor system is representation agnostic; we need to plug-in some representations and metrics to handle this specific pose representation. Specifically:

distance metric — the χ^2 distance between the overlapping parameter sets in the nodes (the pose-graph is highly connected, thus this distance metric is less important than in other motor systems). Specifically for all of the parameters N that are present in two poses a and b we have a distance

$$d_{a\to b} = \sum_{i=0}^{i=N} \frac{|a_i - b_i|}{|a_i + b_i|}$$

time metric — each *ad hoc* parameter has a well-known range, and a well-known time-scale for traversing this range, the time metric is given by the average of each of the times given by these ranges.

interpolator — the interpolator for poses is a bundle of interpolators for each individual parameter, each parameter parameter has a "bias" $\beta \in [0, \infty]$ that modifies these linear blends between two values a and b to be $v = b\alpha^{\beta'} + a(1-\alpha^{\beta'})$ where $\beta' = \beta$ if $|a| > |b|$ otherwise $\beta' = 1/\beta$. This means that the interpolator will err on the side of smaller values of v for $\beta < 1$. This, and the ability to interpose nodes into the pose-graph to deal with the special cases, seemed to give enough control to avoid situations where a linear, independent blend of a great many rendering parameters produced unpleasant intermediate renderings. Finally, *Loops's* motor programs are always interruptible, all blenders are capable of taking the rendering parameters set from any intermediate state.

In addition to causing the rendering style of the creature to move around the pose-graph, actions also choose to modify what the body of the creature is connected to. The body is a set of ordered lists of line segments, with this creature at the head of each list. To keep things simple (and stochastic), the actions which modify the connectivity of the body fall into three different operations:

delete(*segment distribution*) — deletes a line segment drawn from a distribution;

change(*point distribution, segment distribution*) — changes a segment to connect to a point

add(*point distribution, segment distribution*) — adds a point after a particular segment

An action might perform these operations regularly until some condition is met — for example it might delete(*anything*) until there are no more connections left; or it might add(*any_opposite_hand, shortest_list*) until it is connected to 5 other points.

We then construct a vocabulary of distributions. **Point distributions:** *anything* — any point; *any_opposite_hand* — any point on the opposite hand; *corresponding_opposite* — the point corresponding to this on the opposite hand; *correct_hierarchy* — the points above and below in the "correct" hierarchical skeleton of the hand; *across_hand* — the points at the same level on nearby fingers; *connected_amount* — proportional to how many connections a point has; *is_connected_to_me* — only points that are connected to this point; *nearby* — points close by. **Segment distributions:** *longest, shortest, furthest, closest* — each over the length of the segment chains in spatial distance or numerical segment length. Fuzzy binary and unary boolean operations "or" (a+b), "and" (a*b), "not" (1/(1+a)) were also created such that these can be put together.

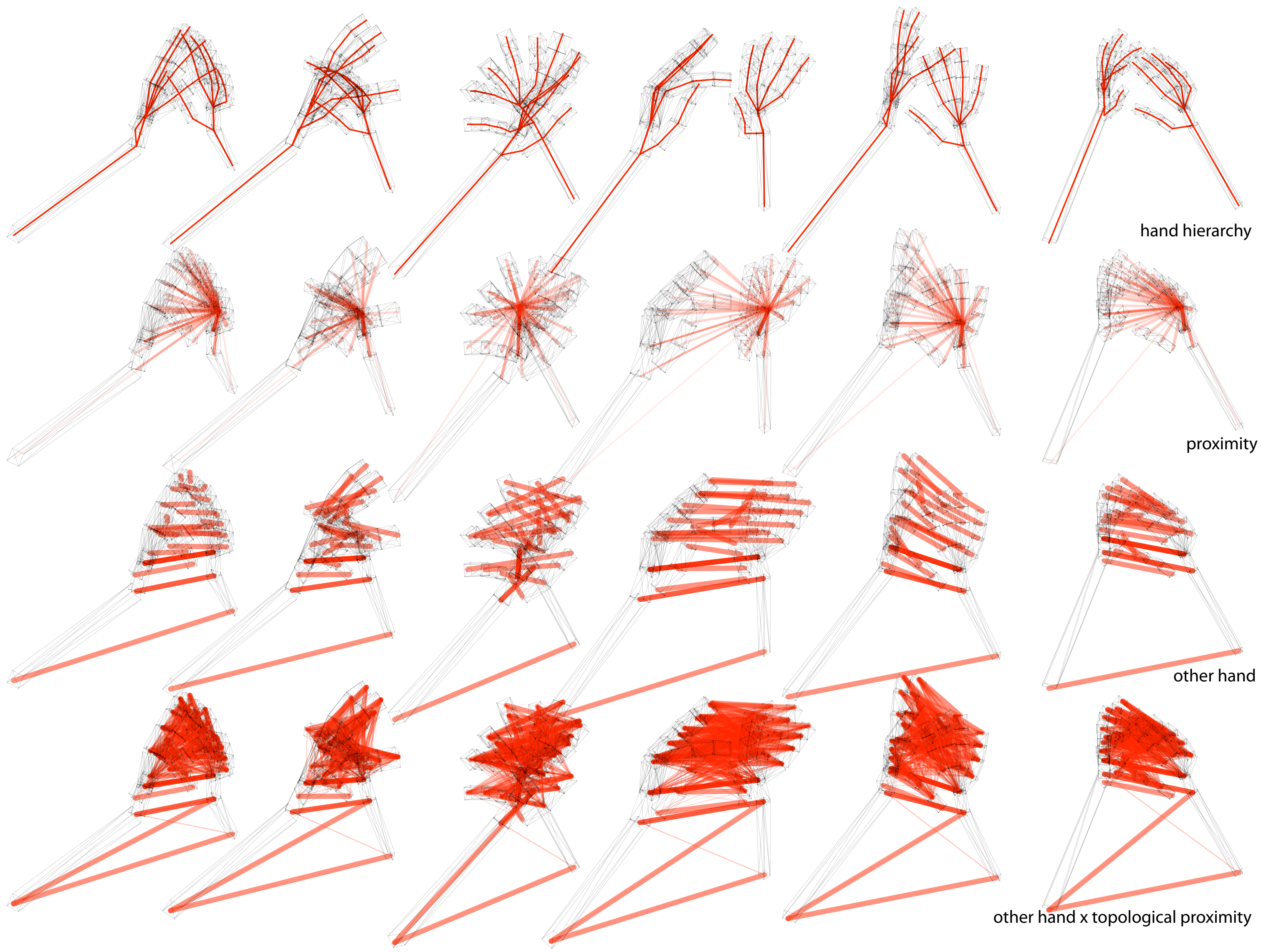


figure 24.
This diagram shows a small sampling of the connectivity metrics distributed across successive frames of motion capture animation.

Signals and expectations

Signals are named, vector values that propagate away from points inside hidden fluids — one independent fluid for each vector signal. *Loops* chooses to model the fluid by storing a highly down-sampled history of point positions and a sampled history of signals sent by each point.

> The signal s present at a particular location p for M signaling markers at positions $p_m(t)$ at time t_0 is approximately:
>
> $$s_p = \sum_m \int s_m(t_0 - |p - p_m(t)|\, v)\, dt$$
>
> where v is the speed of propagation and $s_m(t)$ is the signal sent by marker m at time t. In practice, of course, the integral is replaced by a sum over the (highly) down-sampled history of each point.

Signal transmission has a refractory nature to it — a signal sent constantly gets used up and fades to a low value, so the fluid model is more efficiently modeled as a set of sources, as above and as a general background. This sparsifies the sum over markers above.

This weight average is fed into the action systems of each creature and in the majority of cases, receiving this signal causes the creature to perform a similar action. However, this is not necessarily a constant cause of homogeneity inside the colony — it is easy to author situations using the refractory mechanism that are homogenous but highly unstable, ripe for change. And when this change occurs isolated pools of different behaviors spread throughout the colony and compete for space. It can be some time before homogeneity arises again. These behavior-activation patterns are reminiscent of the percolation behavior of porous solids in physics, or the simulation of "forest-fires" in experimental mathematics, *page 342*.

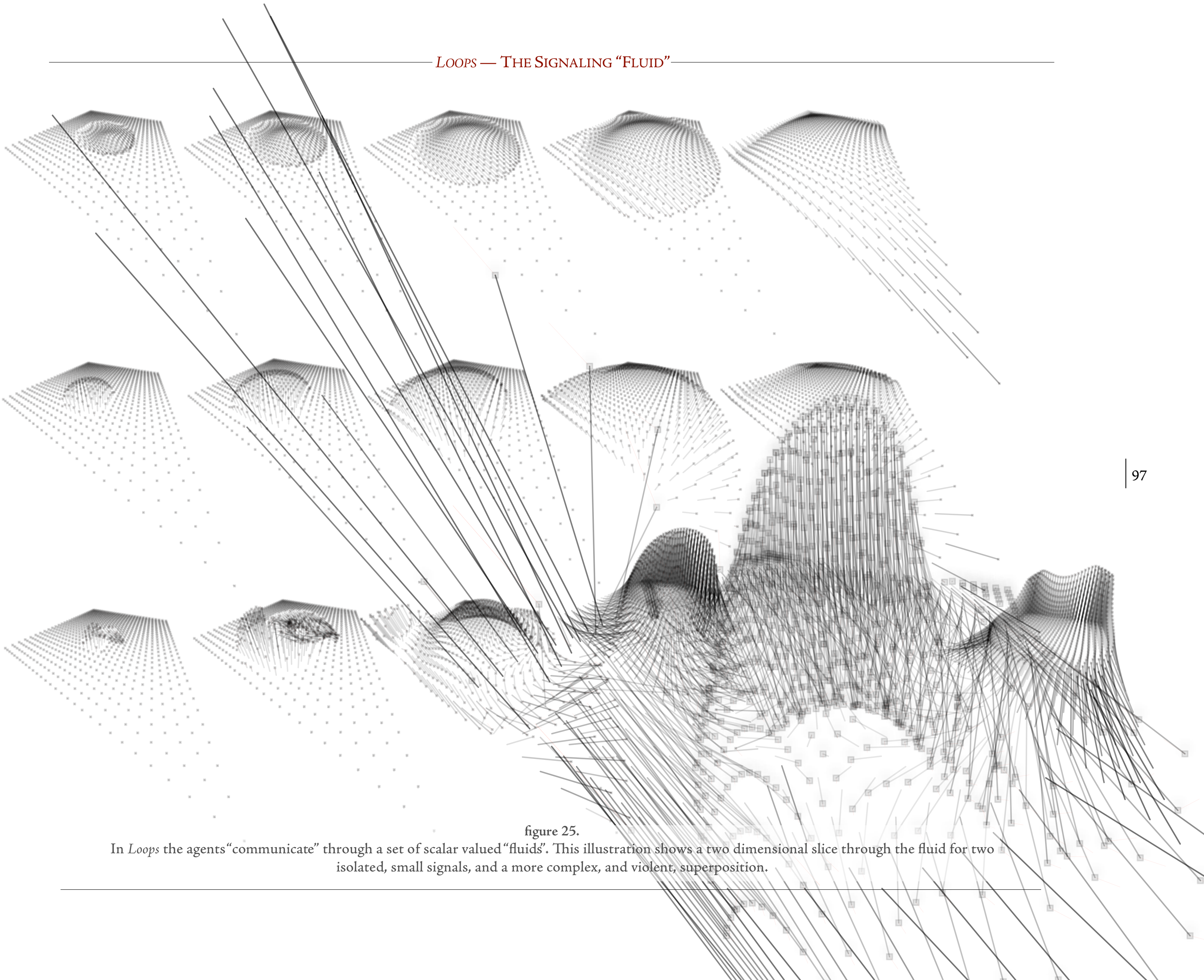


figure 25.
In *Loops* the agents "communicate" through a set of scalar valued "fluids". This illustration shows a two dimensional slice through the fluid for two isolated, small signals, and a more complex, and violent, superposition.

In addition to forming this weighted average, we can also form a weighted radius:

$$r_p = \max \left| s_p - \sum_m \int s_m(t_0 - |p - p_m(t)|\, v)dt \right|$$

This radius is used as the basis for the *expectation* models of the creatures. Integrating across all of the signals, normalized over a long time period, creatures can observe ongoing, conflicted change nearby in the colony. One of the most important, non-action, signals propagated through the hidden fluid is the acceleration of the points themselves. Therefore, the motion of the hands, and in particular their unexpected motion, seeps into the action selection of the points that perform the motion.

Naming

The final set of behaviors and behavioral parameters that made it into the creatures (some 30 action-tuples) are pushed around by a running script which consists, in essence, of a series of these named states. The act of recalling a *name* either resets an adapted parameter to a particular value, resets a equilibrium point in a drifting parameter, or modulates the value of an action tuple to make it more or less likely to fire.

But what does a name refer to? and should the author of an agent have to know? Much ambiguity remains in the single act of naming — does our new label "forest fire (white)" refer to these refractory periods, or the values of those signaling behaviors? In *Loops* the ambiguity is reduced by collecting multiple examples (and remember, for many named states in the colony, we have up to 42 examples for any particular snapshot). By using the consistency between examples to modulate the effect of the saved parameters, when they are recalled, these multiple examples help articulate what it is that we are specifically interested in, what it is that the artists are in fact naming. Those parameters which show little variety throughout all the examples, upon recall, act forcefully upon the creatures' action systems; those that show no consistency have no force upon reapplication.

K-means is a standard unsupervised clustering algorithm — I enjoy the presentation in **C. M. Bishop**, *Neural networks for pattern recognition*, Claredon Press, Oxford. 1995.

This formulation of the use of the Bayesian information criterion is after **D. Pelleg** and **A. Moore.** *X-means: Extending K-means with efficient estimation of the number of clusters.* In Proceedings of the 17th International Conference on Machine Learning, pages 727–734. Morgan Kaufmann, San Francisco, CA, 2000.

One must be a little careful as to how this"consistency" is calculated if we are to fully exploit the information contained within a potentially heterogeneous set of examples. Rather than using the spread of a value, or its standard deviation we repeatedly cluster using a simple k-means clusterer with k=(1..4). We choose our"spread" to be the maximum of the Bayesian information criterion (BIC):

Given a particular clustering C of the (q-dimensional) points $\{p_i\}$ with $i = 1 \ldots n$

$$\mathrm{BIC}(C|\{p_i\}) = \mathcal{L}(\{p_i\}|C) - \frac{k(q+1)}{2}\log n$$

we take the maximum likelihood estimated, log-likelihood $\mathcal{L}(\{p_i\}|C)$ assuming k spherical clusters with centers μ_i each with n_k points:

$$\sum_{c=1\ldots k}\left[-\frac{n}{2}\log(2\pi) - \frac{n \cdot n_k}{2}\log(\hat{\sigma^2}) - \frac{n-k}{2}\right]$$

with (point p_i belongs to a cluster with center $\mu_{(i)}$):

$$\hat{\sigma^2} = \frac{1}{n-k}\sum_i \left\|p_i - \mu_{(i)}\right\|$$

We can then compute a raw consistency measure just from the"error" of the highest scoring clusterer, here we set this to be $1/\hat{\sigma}^2$.

We then allow each cluster to act separately on the action system weighted by the distance from the current parameter to the cluster. These weights are nor-

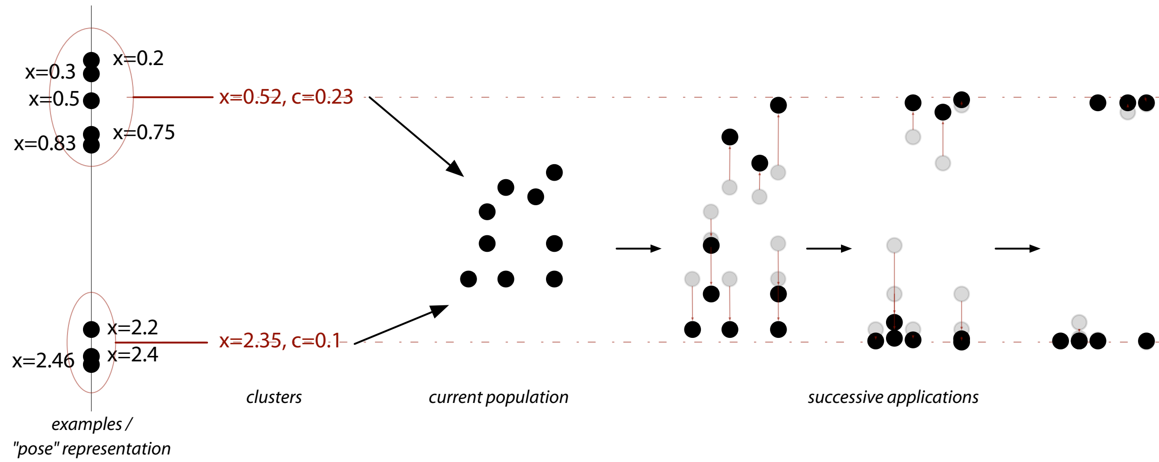


figure 26. Multiple examples inside the pose representation are clustered. During "recall" or pose-interpolation these clusters act on the creature or the colony separately, to reintroduce heterogeneity.

malized over all clusters in this example, but multiplied by our consistency measure (normalized over all clusters for this value in the database):

Given a cluster C_i with mean μ_i acting on a attribute vector v the iteration is:

$$v \leftarrow v + \alpha \frac{\sum_{i=1}^{k} \left\{ (\mu_i - v) \cdot e^{(\mu_i - v)} \right\} \cdot Z / \hat{\sigma}_i^{\,2}}{\sum_{i=1}^{k} e^{(\mu_i - v)}}$$

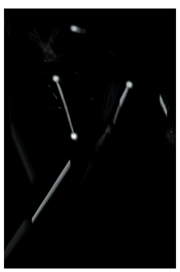
where Z is max $1/\hat{\sigma}^2$ for this value in the database.

There are two desirable results of this scheme: parameter bundles that have, for example, two clearly defined examples for the sample parameter are not penalized as "inconsistent" and thus are weakly recalled if these exemplars are widely separated; secondly, such consistent, but heterogeneous example sets work to increase (and ideally, restore) the heterogeneity of the colony when applied.

These names appear in *Loops* in two places. Firstly they are the bundles of rendering parameters stored in the creatures' pose-graph motor systems. They are recalled by the actions, and smoothly transitioned to by the motor system as described above. Early in the production of *Loops*, these bundles of parameters are created by the hand exploration of the rendering space; later in the production they are sampled and stored from the colony using these techniques. I see this shift from manual exploration to exploration enabled by the processes, if not provoked by the processes, and supported by our agent metaphor, to be a significant hybridization of method in *Loops*. Secondly, names that refer to non-motor parameters — the running actions, the coupling from actions to signals, and the filter dynamics of the refractory system — are assembled by hand, into a looping script which declares at what time what named states act on the colony and with what magnitude (α above). This is created by the artist's loosely distributing change and contrast throughout the 18-minute cycle of time.

In any case, a rather odd thing has happened here. We can recast this scripting view of all of the creatures' action systems as a basis representation for a "body" for a super-agent motor system. Some of the symptoms that are hallmarks of a motor-system like solution appear at this level, *69*: we are interested in manipulating the flow of time through an otherwise constrained set of examples (the *sequencing* of animation); there are constraints that are specified in terms of how this flow can occur that cannot be made ahead of time (the *constraints* of *contents*). Although the "script" for *Loops* ultimately consisted of a single chain (in an endless loop) of "actions" driving this action-system-motor-system, this layering of representational style hints at future conceptual possibilities and technical implementations. This is our first inversion and embedding of the perception-action-motor decomposition within itself — there will be others, and when the time comes to revise the agent toolkit, these inversions will be expected, *page 195*.

throughout the script there are references to terms such as "**xRay**" or "**amoeba**". These are names that the artists used to talk about the basic stylistic vocabulary built for the piece. They refer to behavioral tendencies, connection topologies and/or rendering styles. These common labels became increasingly important as the piece's stylistic vocabulary developed.

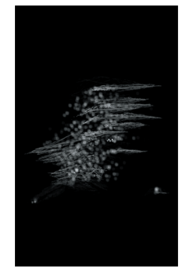


the creatures are responsible for showing how they are connected to other points. Sometimes they choose to connect themselves to points that are make sense in a traditional joint hierarchy. However, they can choose to produce complex 'cat's cradles' or sparse points.

```
public class LoopsScript
        implements Updateable
{

        Script s;

        public LoopsScript()
        {

                s = new Script();
                s.new LoopingRealTimeBase
                        (mins(10.6), true);

                // cats cradle
                s.new Event( 0.1 ).add(new String[]{
                        "s_pureCatCradle=100",
                        "s_cameraTime=0.6",
                        "s_timeFlow=2",
                        "pointTrans=0.3"
                });


                // a little of "xray"
                s.new Event( 30 ).add(new String[]{
                        "s_xrayContext=5"
                });

                // back away (make the transition tentative)
                s.new Event( 35 ).add(new String[]{
                        "s_xrayContext=0"
                });

                // mixed state — some "xray", some "cat's cradle"
                s.new Event( 40 ).add(new String[]{
                        "s_xrayContext=5",
                        "s_pureCatCradle=5"
                });

                //force a transition into "xray"
                s.new Event( 45 ).add(new String[]{
                        "s_xrayContext=100"
                });

                s.new Event( 46 ).add(new String[]{
                        "s_pureCatCradle=0"
                });

                // thin out density and show points
                s.new Event( 45 + 30 ).add(new String[]{
                        "s_nothing=10000",
                        "s_forwardSampling=0",
                        "s_accPointSize=1",
                        "pointTrans=0.2"
                });
```

```
// scribble (by sampling motion forward in time)
s.new Event( 45 + 30 + 25 ).add(new String[]{
        "s_onlyPoints=0",
        "s_forwardSampling=10",
        "s_xrayContext=0"   //???
});

// complicate matters by introducing some "whiteGia"
s.new Event( 45 + 30 + 45 ).add(new String[]{
        "s_whiteGia=20",
        "s_nothing=1",
        "s_accPointSize=0",
        "pointTrans=0.0001"
});

// move camera towards hands
s.new Event( 45 + 30 + 45 + 20 + 20 ).add(new String[]{
        "s_forwardSampling=10",
        "s_nothing=0",
        "s_cameraTime=0.47"
});

// transition to "whiteGia" complete
s.new Event( 45 + 30 + 45 + 20 + 25 ).add(new String[]{
        "s_whiteGia=100",
        "s_nothing=0",
        "s_forwardSampling=2",
        "s_timeFlow=1"
});

// propagate force messages between creatures
s.new Event( 45 + 30 + 45 + 20 + 25 + 20 ).add(new String[]{
        "s_doForceFlare=100"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 25 ).add(new String[]{
        "s_doForceFlare=10"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 35 ).add(new String[]{
        "s_doForceFlare=0"
});

// thin out "whiteGia"
s.new Event( 45 + 30 + 45 + 20 + 25 + 45 ).add(new String[]{
        "s_nothing=1",
        "pointTrans=0.2"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 47 ).add(new String[]{
        "s_nothing=0"
});
```

"**whiteGia**" refers to a rendering style that we found reminiscent of the line quality of Giacometti's portraits.

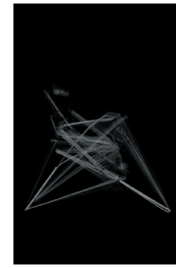


"**forward sampling**" randomly elongates the point creatures bodies along the animation material. The result is often similar to inverted comet trails.

"**force propagation**" refers to the virtual dispersive medium that the point creatures are 'embedded in'. Creatures can inject force into what is in essence a simple cloth simulation, to perturb and fatten the geometry of nearby creatures.

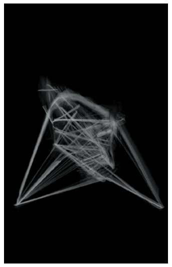
the behavior system paradigm is particularly good at generating hybrid states with some of the colony agreeing on one behavior and others performing some other actions.

```
// scribble (by sampling motion forward in time)
s.new Event( 45 + 30 + 25 ).add(new String[]{
            "s_onlyPoints=0",
            "s_forwardSampling=10",
            "s_xrayContext=0"   //???
});

// complicate matters by introducing some "whiteGia"
s.new Event( 45 + 30 + 45 ).add(new String[]{
            "s_whiteGia=20",
            "s_nothing=1",
            "s_accPointSize=0",
            "pointTrans=0.0001"
});

// move camera towards hands
s.new Event( 45 + 30 + 45 + 20 + 20 ).add(new String[]{
            "s_forwardSampling=10",
            "s_nothing=0",
            "s_cameraTime=0.47"
});

// transition to "whiteGia" complete
s.new Event( 45 + 30 + 45 + 20 + 25 ).add(new String[]{
            "s_whiteGia=100",
            "s_nothing=0",
            "s_forwardSampling=2",
            "s_timeFlow=1"
});

// propagate force messages between creatures
s.new Event( 45 + 30 + 45 + 20 + 25 + 20 ).add(new String[]{
            "s_doForceFlare=100"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 25 ).add(new String[]{
            "s_doForceFlare=10"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 35 ).add(new String[]{
            "s_doForceFlare=0"
});

// thin out "whiteGia"
s.new Event( 45 + 30 + 45 + 20 + 25 + 45 ).add(new String[]{
            "s_nothing=1",
            "pointTrans=0.2"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 47 ).add(new String[]{
            "s_nothing=0"
});

// transition to a "forest fire" based dynamics
s.new Event( 45 + 30 + 45 + 20 + 25 + 50 ).add(new String[]{
            "s_nothing=100",
            "s_forestFireOne=1",
            "s_cameraTime=0.6",
            "s_randomPointSize=0.0",
            "pointTrans=0"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 53 ).add(new String[]{
            "s_nothing=0"
});

// a delicate, transient state
s.new Event( 45 + 30 + 45 + 20 + 25 + 60 ).add(new String[]{
            "s_whiteGia=0",
            "s_catCradle=10",
            "s_xrayContext=1",
            "s_randomPointSize=0.0"
});

s.new Event( 45 + 30 + 45 + 20 + 25 + 80 ).add(new String[]{
            "s_cameraTime=0.47"
});

// "message passing" between randomly flashing points
s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 ).add(new String[]{
            "s_message=40",
            "pointTrans=0.4",
            "s_catCradle=0",
            "s_xrayContext=0",
            "s_randomPointSize=0.4",
            "s_cameraTime=0.47",
            "s_timeFlow=0.2"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 30 ).add(new String[]{
            "s_randomPointSize=0.0",
            "pointTrans=0.4",
            "s_timeFlow=1"
});

// a complex heterogeneous state — Large Glass?
s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 45 ).add(new String[]{
            "s_tendril=55",
            "pointTrans=0.00001",
            "s_xrayContext=1",
            "s_message=0",
            "s_timeFlow=0.5",
            "s_cameraTime=0.6",
            "s_randomPointSize=0"
});
```



the way in which the point creatures adapt their geometry to indicate how they are 'connected' to other points changes throughout the piece. One of the earliest styles we built was the "tentative tendril" growth style, where points seem to be gently seeking nearby points in the hand.

by changing how gradually or suddenly new behavioral tendencies are introduced into the creatures by the script we can modify the abruptness of the transition. If we quickly force a behavioral tendency to have a very high value we *startle* creatures into revaluating their behaviors. But by gradually introducing new behavior we can create hybrid and "indecisive" states into the colony.

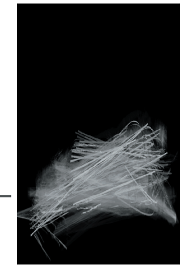


"**nameBug**" refers to a dense, nested connection topology between points that was introduced by a simple mistake in code. Once the mistake was traced and corrected we rebuilt the style.

"**forest fire**" message propagation refers to a complex extension of the "force propagation" used earlier. Instead of passing force into a simple physics simulation, points pass messages of behavioral tendency. This creates a deliberately brittle positive feedback system. Behaviors change between points in a way similar to how fire spreads in a forest. These complex behavioral dynamics were extensively simulated in isolation and could be visualized while the piece was running.

```
// which we slow down for
s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 45 + 14 + 81 + 20 )
        .add(new String[]{
        "s_timeFlow=0.2"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 45 + 14 + 81 + 30 )
        .add(new String[]{
        "s_timeFlow=0.5"
});
s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 45 + 14 + 81 + 40 )
        .add(new String[]{
        "s_timeFlow=1.5"
});

// startling transition into "name bug" connection topology
s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 45 + 14 + 91 + 45 )
        .add(new String[]{
        "s_bugName=1000",
        "s_amoeba=0",
        "pointTrans=0.2",
        "pointWhite=1",
});
s.new Event( 45+30+45+ 20 + 25 + 80 + 45 + 45 + 14 + 91 + 45 +10 )
        .add(new String[]{
        "s_timeFlow=0.2"
});
s.new Event(45+30+45+ 20 + 25 + 80 + 45 + 45 + 14 + 91 + 45 + 15 )
        .add(new String[]{
        "s_timeFlow=1"
});

// thin out density
s.new Event(45+30+45+20 + 25 + 80 + 45 + 45 + 14 + 101 + 45 + 20 )
        .add(new String[]{
         "s_nothing=10",
        "s_timeFlow=0.2" // ?
});
// thin out density
s.new Event(45+30+45+20 + 25 + 80 + 45 + 45 + 14 + 101 + 45 + 25 )
        .add(new String[]{
        "s_nothing=0"
});
// complex forest fire with densely connected graph
s.new Event( 45+30+45+20 + 25 + 80 + 45 + 45 + 14 + 101 + 45 + 30 )
        .add(new String[]{
        "s_forestFireOne=10",
        "s_bugName=2",
        "s_tendril=0",
        "s_timeFlow=1",
        "s_cameraTime=0.47"
});
```

```
        s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 45 + 14 + 101 + 45
    + 35 ).add(new String[]{
                "s_forestFireOne=100",
                "s_forwardSampling=100",
                "s_bugName=0",
                "pointTrans=0.05",
                "s_timeFlow=1",
        });

        // move camera in
        s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 45 + 14 + 101 + 45 + 35
    + 40 ).add(new String[]{
                "s_cameraTime=0.6"
        });

        // starting transition into tendril growth, with whatever
        // stylistic parameters happen to be there
        s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 45 + 14 + 101 + 45 + 35
    + 60 ).add(new String[]{
                "s_forestFireOne=0",
                "s_tendril=100"
        });

        s.new Event( 45 + 30 + 45 + 20 + 25 + 80 + 45 + 45 + 14 + 101 + 45 + 35 +
        60 + 30 ).add(new String[]{
                "s_pureCatCradle=100",
                "s_tendril=0"
        });
    }
}
```



we also can change the speed with which we play out the underlying motion capture data. For example here we craft a moment of calm surrounded by frenetic activity.

we do not return at the end of the script to exactly the same place (behaviorally) as we started from. The next and subsequent times through this script (which is looped) the behavior and choices of the creatures will be slightly more complex, slightly less clear cut and creatures respond to residual behavior in the system. Thus there are behavioral timescales much longer than the length of this script.

Rendering Loops

I conclude this overview of the *Loops* installation with a description of the graphical rendering constructed for the bodies of the agents — which at the time offered a unique exploration of what the computer graphics community would refer to as the "non-photoreal".

The material to be rendered is given directly by the point-line segment level description. These line segments are in fact descriptions for splines that originate at the point agent's position but use other agents as control points. It is on these lines that the *Loops* agents' "rendering parameters" act. These lines are interpreted using one form of parametrically controllable spline — the so-called tcb (tension, continuity, bias) spline family. Roughly speaking these parameters control, per control node, the sharpness (t), the "loopiness" (c) and the asymmetricality (b) of the line. The distribution of tcb values along the lines then marks an important class of rendering parameter.

These splines, in turn, are used to deform predefined geometric meshes that span the space of smooth, rough, spiky — the position inside this blend space forms another parameter that controls the appearance of this agent's line.

Prior to reaching the drawing surface, screen-space noise is added to the position of each vertex. The parameters (amplitude and direction) of this noise are not specified directly as a rendering parameter. Rather, their couplings to the signal-propagation layer are specified. This offers a back door for the action selection of points to be visualized as sweeping across the graphical representation of the colony.

These meshes are alpha-composited into the screen as transparent geometry. And the multiple overlappings of the randomly perturbed geometry add considerable texture to the drawn line. However, this texture is entirely controlled by the geometry and, even instantaneously, offers a genuine patina of process to the drawn material.

Finally, the frame-buffer on which *Loops* is drawn exhibits its own, very simple, memory of process. Rather than, as is typical in computer graphics, clearing the screen prior to each new frame, the previous frame in *Loops* is dimmed slightly and the new frame drawn on top of its fading trace. The result is an accumulation of geometry-driven textural complexity.

Calculating the spline-based distortions of the blended mesh was sufficiently complicated to occupy much of the processor power present on the high-end commodity hardware available in 2001 when this piece was constructed. Needless to say, Moore's "Law" has turned this aspect of *Loops's* renderer into an altogether more trivial computational load. The mechanism behind the distinctive appearance of *Loops* was reconsidered for each subsequent work and its "hand drawn" aesthetic can be felt in even my most recent work.

Gordon E. Moore's "Law" states that the complexity of an integrated circuit available for a fixed cost doubles every 18 months. For example: http://en.wikipedia.org/wiki/Moores_Law

The longevity of this hand-drawn "look" in my work is not motivated by the desire to display technical virtuosity, nor the delight in a perverse re-appropriation of the hardware designed for the photo-real. Rather, it stems from the importance of the sense of effort, the sense of mistake and subsequent correction, and the sense of being trapped within and exploring a finite world of possibilities. Even in the most recent dance piece *how long...* as we move from a viewpoint of agent-as-computing the work to agent-as-seeing the work, this hand-drawn referent remains motivated by that work's *notational* concerns. The effort, the uncertainty and ultimate transience of the hand-drawn, the sketched, remains.

Of course, this geometrically controlled emergence of texture was reinvestigated using more computationally intensive strategies beyond that of simply not clearing the frame-buffer "properly" between successive frames. And my fascination with the gestural connotations of the drawn line motivated a balancing counter move back towards the "photo-real" in the work *22*.

3. Concluding remarks — authorship and emergence

The simplest form of adaptation that can occur in this system is one that modifies the "internal value" of the action tuple. Lower the value, and the probability of it firing, all other things being equal, decreases. This kind of adaptation, provoked by simple, hand-reinforcement, occupies a kind of middle ground — both technically and temporally — in a chain of possible adaptive processes that shaped *Loops* while it was being made. For *Loops* was made as a collaborative work, and in this particular case a collaboration with non-programmer artists. After assembling a certain confidence in our materials, our methodology began with running a small-scale version of the work and tuning it, and growing it. This small-scale version started by using a limited amount of motion-capture material (to help the us maintain our bearings) and limited sections of the action system — over time, pieces of action system that had been worked on extensively were pieced together and the system opened up to more motion material.

Initially, most of the tuning took place on the smallest, least "process", and most "direct" level: the appearance of the creatures, and the shape of the blend spaces that their rendering parameter-based bodies traversed; by the end, most of the tuning was devoted to large scale signaling interactions of the colony. At each level (and there were almost always more than one being worked on at a time), there was a cycle of exploration and adaptation succeeded by naming and verification that there was a consensus of reproducibility between the artists and the colony — that everyone, including the creatures, agreed on the name and what it was that was named.

The list of "adaptive levels" was quite long and reflects, I believe, the depth of collaboration achievable using this stack of adaptation / persistence: **rendering parameters** for the creatures were deliberately altered and new example parameter sets were named and injected back into the blend space available for all creatures; **connectivity tendencies** were assembled and named; **actions** were added into action systems of creatures (sometimes, for experimental purposes to the creatures associated with one hand) and named; **behavioral decisions** were reinforced (and negatively-reinforced), reward signals delivered to the entire colony, to a particular hand, to a set of creatures exhibiting a particular behavior, or, more likely, to a set of creatures exhibiting a particular rendering style; **behavioral configurations** were named, including the preferences created through reinforcement and the internal parameters of the refractory and expectation mechanisms.

Each of these adaptive levels forms an intricate emergent structure; but each pairs a downward specifying *force* on this upwards, emergent, untamed *potential.* Upwards — rendering parameters, although manipulated by hand are constantly being blended together and juxtaposed by the creatures' multiple motor systems; downwards — the sampling, storage and editing of new parameter-sets back into the vocabulary of the colony. Upwards — a basic set of connectivity patterns are created, but here, too, the creatures spend much of their time in intermediate states; downwards — the direct modification of the connectivity metrics. Upwards — the interaction of newly added actions with the existing action system; downwards — the sampling of active actions or the hand creation of partially active sets of actions, or the reinforcement of actions and signaling mechanisms and the annotation of this reinforcement into the script itself.

Thus *Loops* represents in miniature the whole argument of the agent-based practice — it offers a framework for organizing navigational and specificational strategies that mine the potential latent in algorithmic systems. Rather than choosing between a rejoicing in the sheer size of the abstract potential developed by fusion of multi-media and digital process (as offered by artificial life) or the hand-tuned system that acts as a refusal of potential (as offered by practices of mapping), the agent offers an alternative path, where algorithmic, formal ideas are permitted their *potential* while the artist is permitted multiple strategies for exerting their *taste*.

Thus the incredible flexibility of the action system, the renderer and the analysis of motion are paired down, even sculpted, interactively by the artists making this work. At each level, points, directions and planes are stored, named and folded back into the work. The first indication that our method was truly "working" occurred when the colony was first exposed to the whole motion-captured performance. *Loops* became a richer work, a surprising work, and yet, simultaneously remained the same work. This stable expansion of a formal idea is often a distant dream of interactive works — far easier is the over-fitting of a piece's parameters to a particular "correct" interaction; far more common is brittle failure in the light of the unexpected. Even today the resulting system is both complex and opaque enough to keep some of its secrets until years have passed — this installation has been touring since 2001, and is booked through until the end of 2005 — and yet was, during its making, controllable enough that these surprises could be captured, assured and incorporated back into the work.

While the specifics of each of the levels themselves are concretely tied to *Loops*, this idea of a stack of such levels is not. Indeed, this layering of freely emergent systems with systems that impose not order or control, but explicitly a navigation or the ability to draw a map, offers us a general alternative model to that of artificial life. The agent-metaphor, together with it telescoping structure of time-scales and self-interactions helps organize how this stack intersects with the artwork's interaction — even if, in this case, the work only interacts with the artists as they are developing it.

We'll note in passing that *Loops*, although it reuses much of the c43 toolkit, occupies a different area of our earlier axial decomposition of action selection techniques. Viewed from outside, from the perspective of the creators of this work, *Loops* "action selects" on two levels: *Loops's* multiple, interacting creatures allows access to the "multiple concurrent actions" domain previously far away from c43. In the work that follows we'll see further efforts to allow the complex "choreography" of simultaneous actions that moves further away from c43's starting point.

I believe that these layered structures are at the core of why the agent-based offers an organizing alternative to the positions of mapping and emergence, in general, for creating interactive artworks. And while *Loops* was created over a very short period of activity, we shall see this argument only growing stronger as the agent enters either dance theater or long-term collaborative art-making.

Loops then, within its limits, is a work that I claim as successful. Successful in the sense that it leads to something — that is, it does not exhaust the potential developed by its seed technical ideas, but rather leaves one with a better sense of the territory of that potential (for just one example, the point and line based bodies appear, re-generalized, in both *Lifelike* and *how long...*); successful in its creation of an authorable yet emergent methodological process; and successful in remaining open to the opportunities of the material that it interacts with. This given, the real question lies in how to expand these "successes" into larger and more complex works. *Loops* had much many attributes in its favor towards these goals. Although collaborative, it was an intensely personal work — it was very much made on our own terms, in our own time — but what happens when the working practice is expanded out to other collaborators (e.g. choreographers) with other time scales (e.g. rehearsals and workshops) and other non-personal, non-constant spaces (e.g. theaters and galleries)? Although complex, *Loops* establishes this complexity by the duplication of simple parts, and as such risks falling into the anonymity I accused artificial life of cultivating. While the transient presence of the human form prevents the singular from disappearing from *Loops* altogether, it was clear at the time of the work's completion that an engagement with a lower number of more complex agents was on the horizon — that *Loops* has deferred, but not solved, the software engineering problems apparent in *alphaWolf*. Seen in this light the "success" of *Loops* promises much but speaks little to these problems. In order to create the next artworks, these issues would have to be addressed.