(TR # 011) 07/26/91

Smith

TECHNICAL REPORT

Generating Random Polygons

Joseph O'Rourke
Mandira Virmani

Department of Computer Science Stoddard Hall Smith College
Northampton, MA 01063 Tel: (413) 585-3858

Generating Random Polygons

Joseph O'Rourke*
Mandira Virmani†

July 25, 1991

1 Introduction

There is a need for a method to generate "random" polygons for testing algorithms that accept a polygon as input. See, for example, the Carleton "Workbench for Computational Geometry"[EKM+90]. A precise definition of what should constitute a random polygon seems elusive, and no attempt will be made here to clarify this notion. Rather we report on a method we developed which seems to give reasonable results.

2 Sketch of Method

The basic idea is simple. To generate a random polygon P of n vertices within a certain bounded region R, start with an arbitrary polygon of n vertices within that region. In our experiments we chose to start with a regular polygon, which is of course as far from random as possible. Now assign each vertex a fixed velocity in a random direction, with a random speed (no greater than some maximum speed). Permit the polygon to deform over time with its vertices moving like billiard balls, stretching the edges between them appropriately.

At all times we would like to maintain two conditions: the polygon is simple (non-selfintersecting), and it remains within the bounding region R. When a violation of either of these conditions is about to occur, we reassign the offending vertex a new random velocity. In effect, the vertex "bounces off" the edge of P or of R as if it were a ball spinning randomly (so that its new velocity is random).

The polygon is permitted to evolve over time until sufficient "mixing" is attained. We make some half-hearted attempts to measure this mixing, but ultimately the success of the method must be judged by the aesthetic appeal of the shape of the resulting polygons.

*Department of Computer Science, Smith College, Northampton, MA 01063, USA. Supported by NSF grant CCR-882194.

†Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, USA.

3 Method in More Detail

We start with a regular polygon P of n vertices of radius 750 inside a bounding circle R of radius 1000. Velocities are assigned to each vertex, with a random speed uniformly distributed in $[0, 200]$. Vertices are indexed from 0 to $n-1$. The position of vertex i is $p[i]$, and its velocity is $v[i]$. A single time unit evolution is accomplished as follows:

Algorithm 1: ONE TIME UNIT
for each $i = 0$ to $n-1$ do
    Move vertex i tentatively to $p'[i] = p[i] + v[i]$.
    if $p'[i] \in R$ and the new polygon is simple
        then (Accept) $p[i] = p'[i]$
        else (Reject) Assign $v[i]$ a new random velocity.

Note that if an attempted move is in violation of the bounding region or of simplicity, it is simply not made: on the next iteration that vertex will move with a new random velocity.

We currently use a brute-force algorithm to determine if a polygon is simple: check if the two new edges, $(p[i-1], p'[i])$ and $(p'[i], p[i+1])$ intersect any other edges. Thus this step requires $O(n)$ time. One can imagine making this step more efficient: perhaps $O(\log n)$ is possible.

Finally, the "One Time Unit" procedure is repeated until the desired mixing is obtained.

4 Examples for $t = 5$

Before showing sufficiently mixed examples, we illustrate the algorithm working for just five time units, on two examples, both with $n = 36$ but started with different random seeds, in Figs. 1 and 2. The trajectories of the individual vertices are evident in these figures. These (and all succeeding) figures should be read bottom to top, left to right. Thus $t = 0$ is bottom left, and $t = 5$ is top right.

5 Examples for $t = 1000$

We now present six different random polygons for several values of n: $n = 36$, 72, 180, 360 in Figs. 3, 4, 5, 6 respectively. In all cases, the polygons evolved from a regular n-gon in 1000 time units. Each of the six for each n differ only in the seed used for the random number generator.

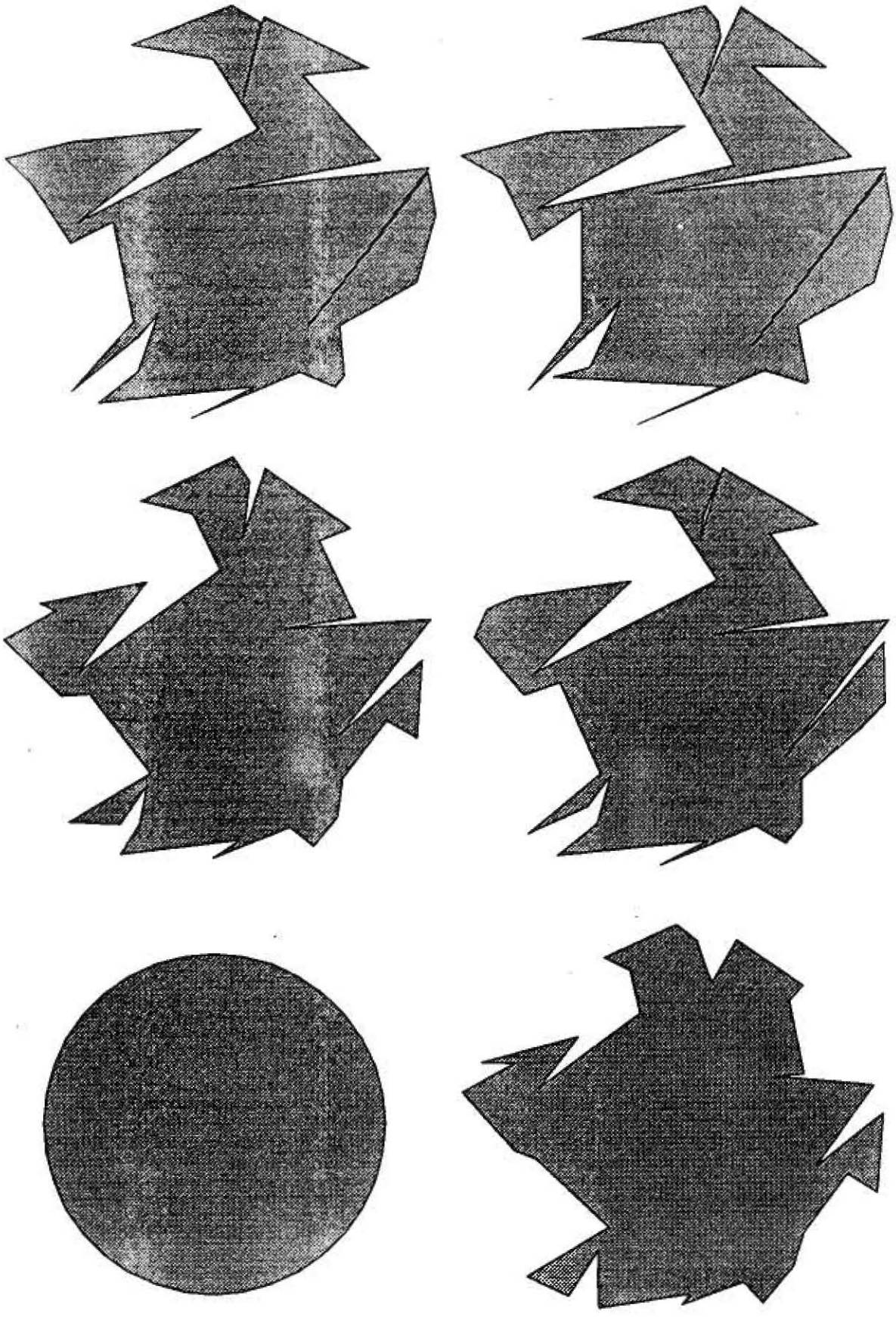

Figure 1: $n = 36$, five time steps (seed = 4).

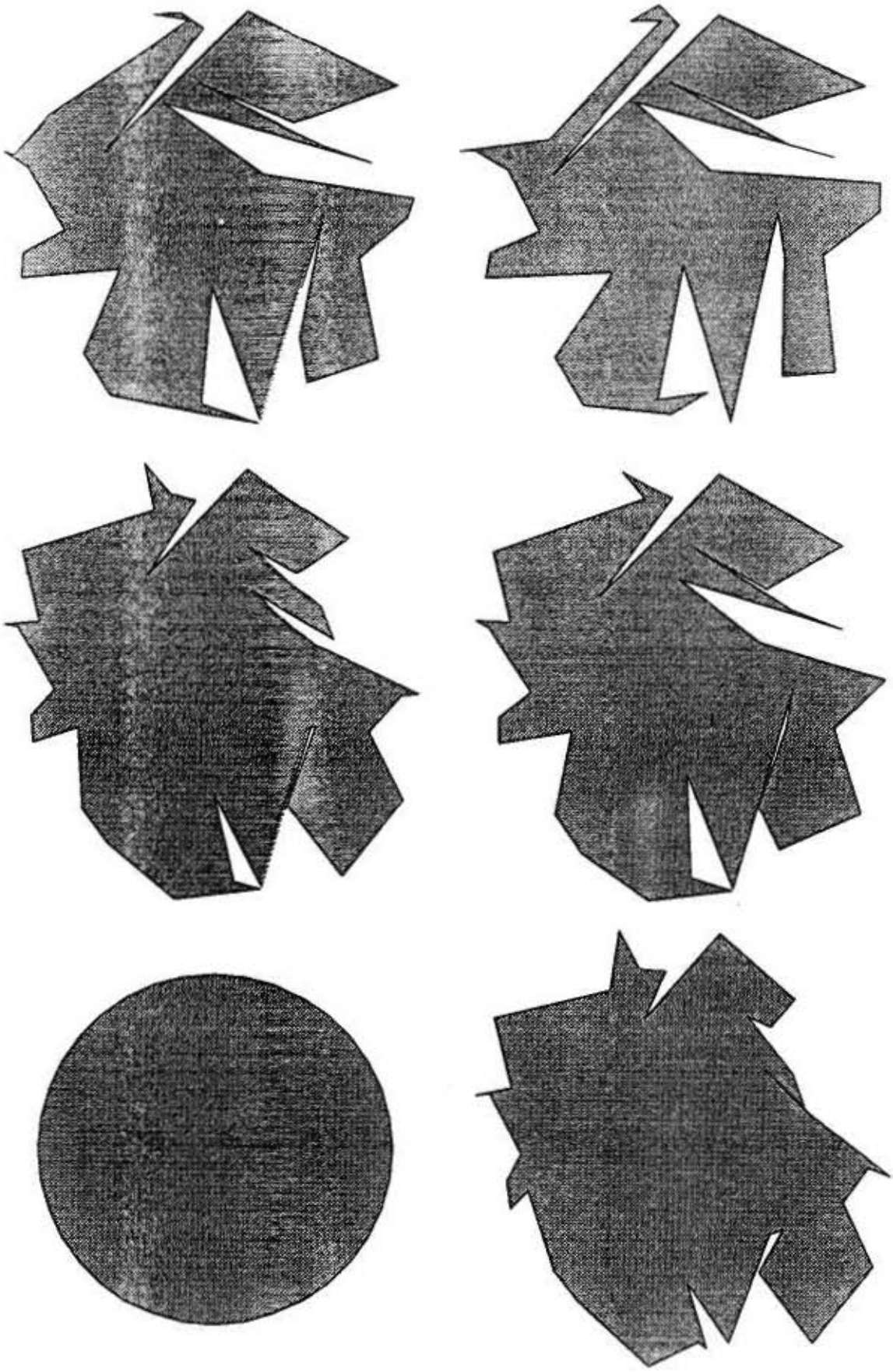

Figure 2: $n = 36$, five time steps (seed = 10).

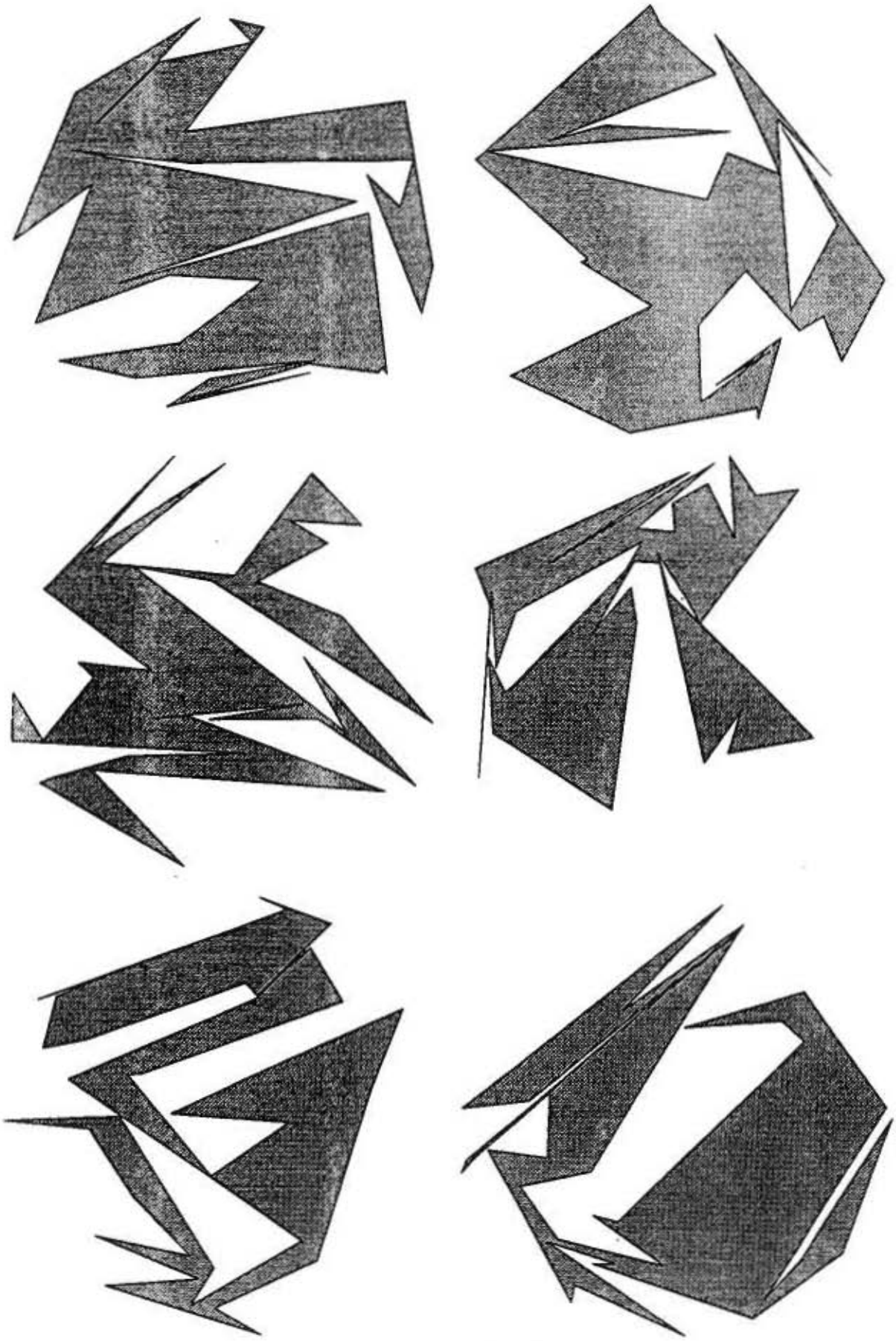

Figure 3: $n = 36$, $t = 1000$, six different random seeds.

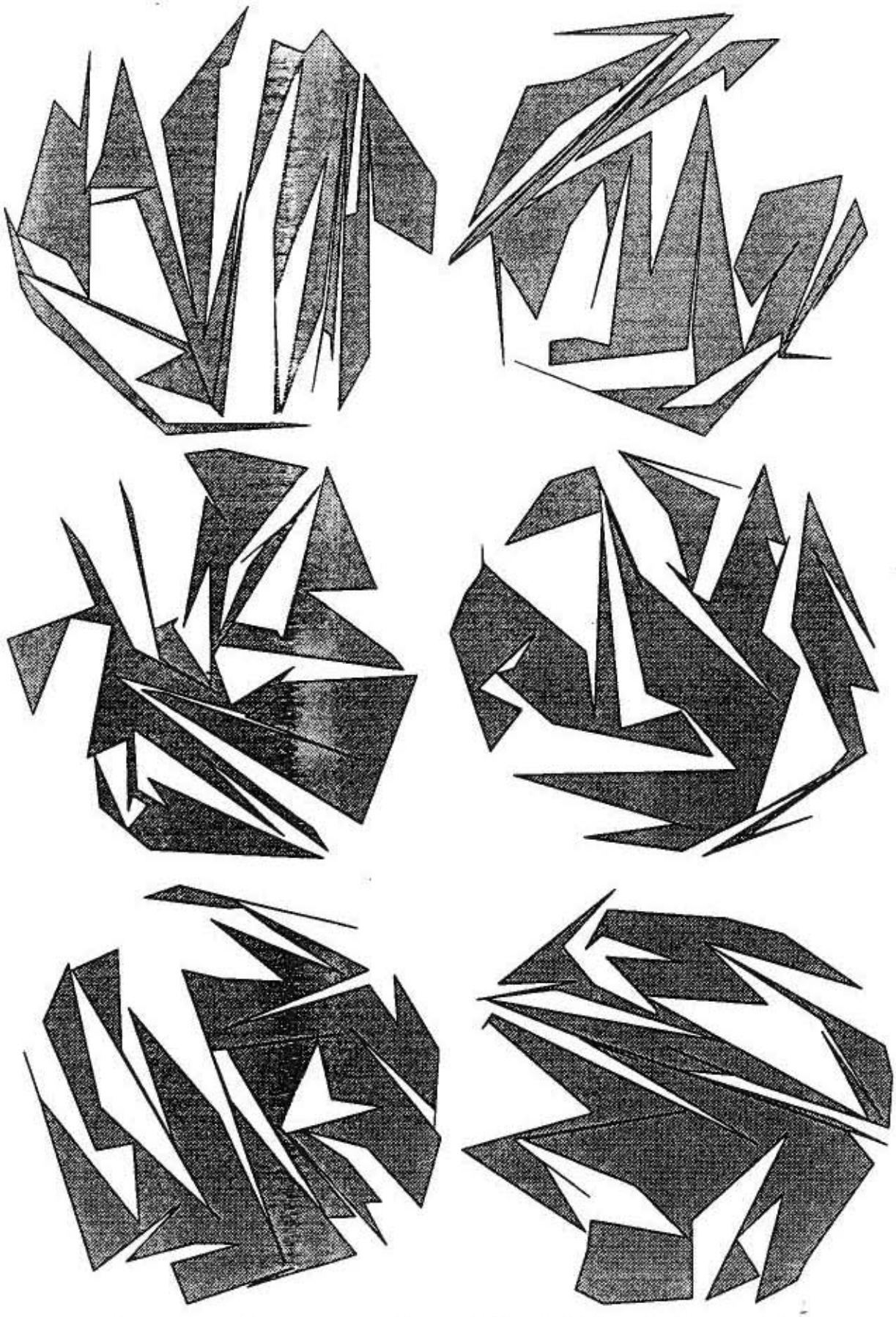

Figure 4: $n = 72$, $t = 1000$, six different random seeds.

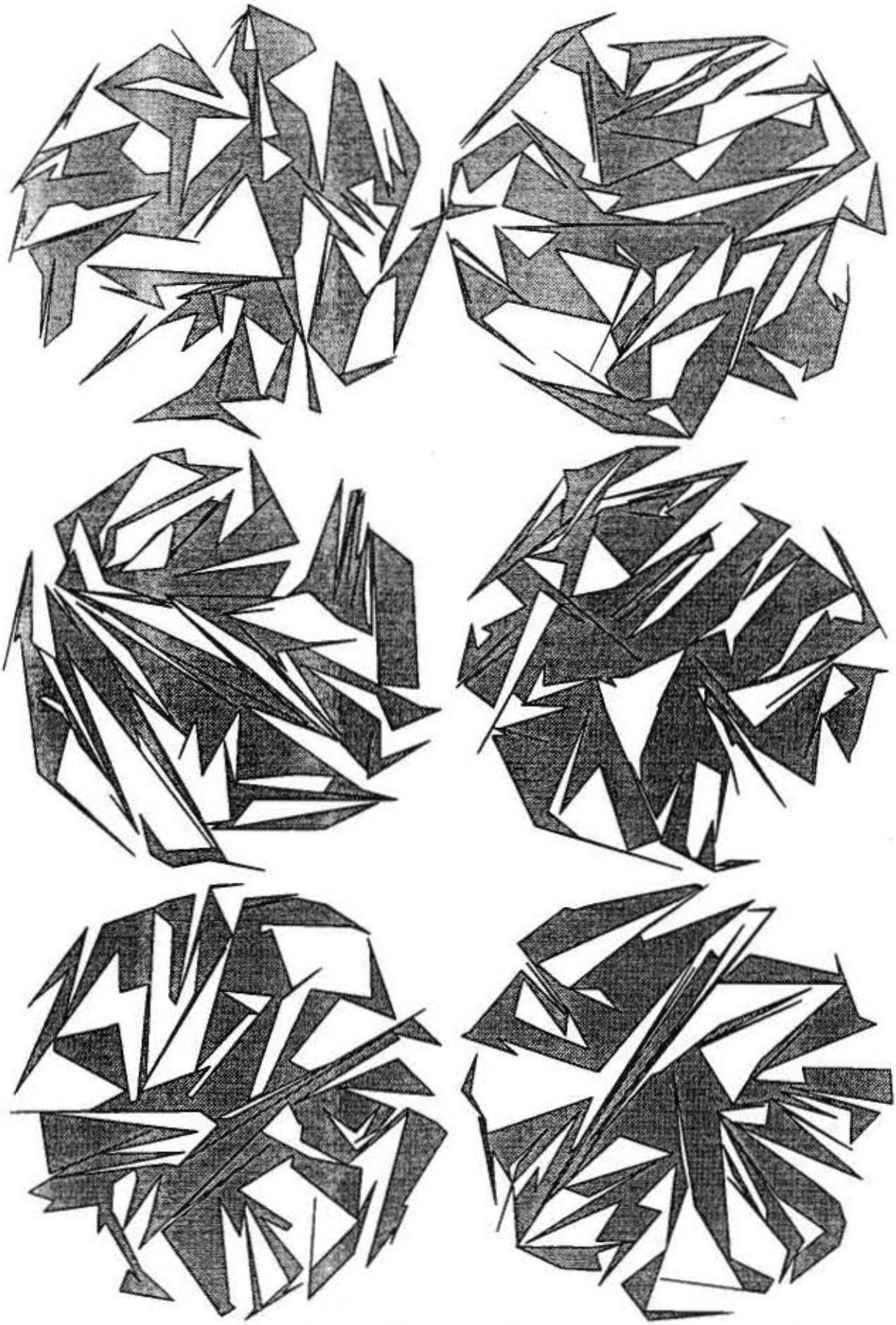

Figure 5: $n = 180$, $t = 1000$, six different random seeds.

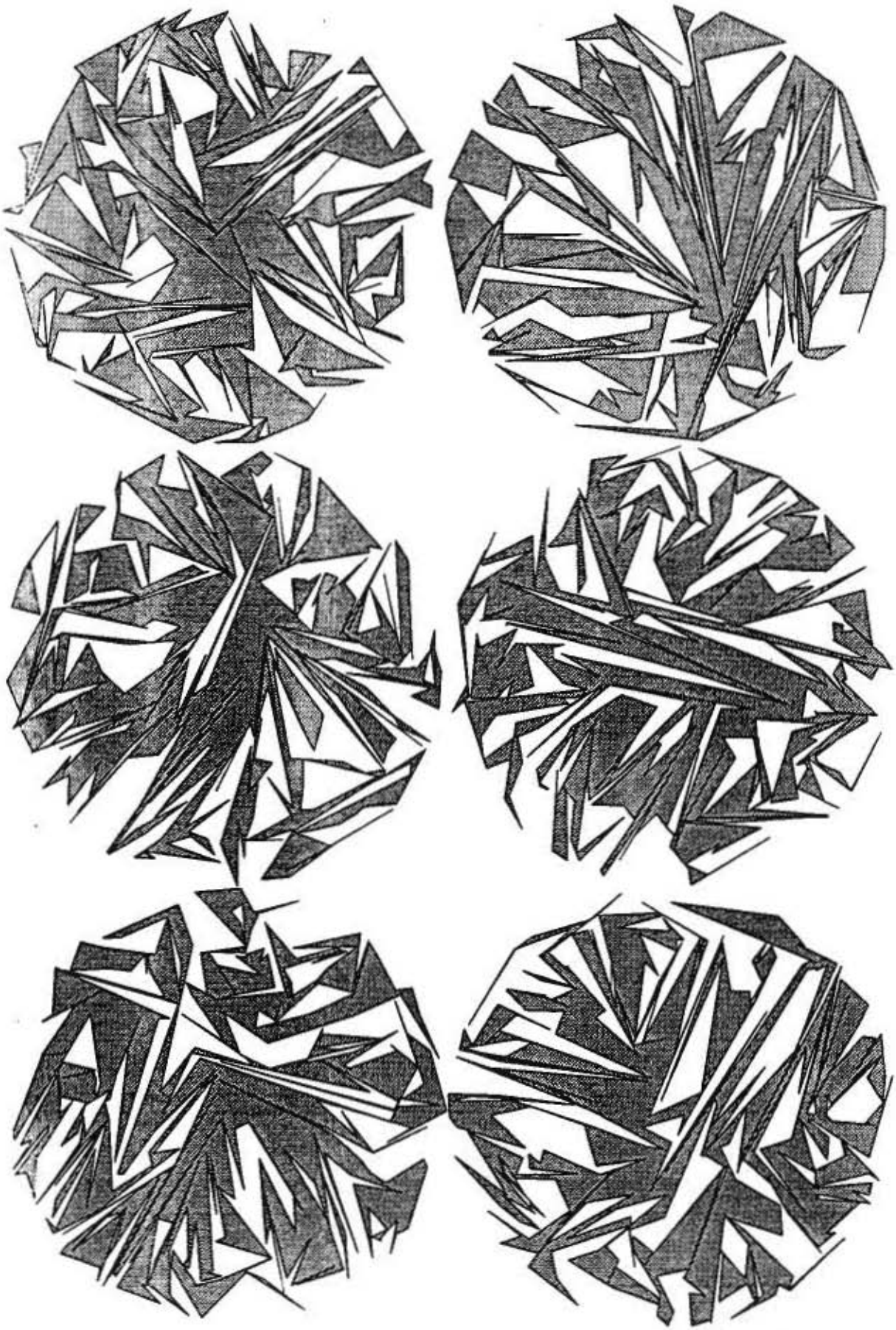

Figure 6: $n = 360$, $t = 1000$, six different random seeds.

6 Measures of Randomness

In an attempt to verify that the polygons in Figs. 3-6 are in fact "random" in some sense, we measured two quantities: the spatial distribution of the polygon vertices[1], and the distribution of the angles of the polygon edges. We call the first the *vertex distribution* and the second the *edge distribution*. Clearly we would like to have both distributions approximate uniform distributions.

6.1 Vertex Distribution

We partitioned the bounding circle R into $k = 8$ equal area concentric annuli, and counted the number of vertices in each annulus. Initially the distribution of vertices into annuli is highly nonuniform, with all n vertices falling into one annulus. We expect that after sufficient mixing, each annulus should contain approximately n/k vertices. Fig. 7 shows the concentric circles overlayed on the random polygons from Fig. 3. We measured the deviation from uniformity χ_v with the Chi-square distribution; see the caption to Fig. 7.

6.2 Edge Distribution

To measure the angle distribution, we partitioned the unit circle into $k = 8$ angular sectors, and counted the number of edges (treated as directed by a counterclockwise boundary traversal) that fell into each sector. Initially we have a perfect distribution, because we start with a regular polygon. We expect that the distribution should stay approximately uniform, with n/k edges in each sector, and again we measured the deviation from uniformity χ_e with the Chi-square distribution. Fig. 8 shows the distribution of the edge vectors for the random polygons in Fig. 4.

6.3 Chi-square Values

For $k = 8$, the relevant Chi-square values and the corresponding percent confidence level that the observed distribution is uniform, are:

90%	2.83
95%	2.17
99%	1.24
99.5%	0.989

In Fig. 9 we show snapshots of the evolution of an $n = 36$ polygon at $t = 400$ intervals up to $t = 2000$ ($t = 0$ bottom left, $t = 2000$ top right), and in Fig. 10 we plot the Chi-square values for both the vertex and edge distributions.[2] Figs. 11 and 12 show similar pictures for $n = 72$, and Fig. 13 shows the distributions for a similar $n = 180$ example (not shown). These examples illustrate that both distributions are uniform

[1] Eli Goodman suggested this.

[2] The point plotted for χ_v at $t = 0$ is not correct in this and succeeding figures: it is even larger, but we moved it down to prevent the vertical scale from being stretched.

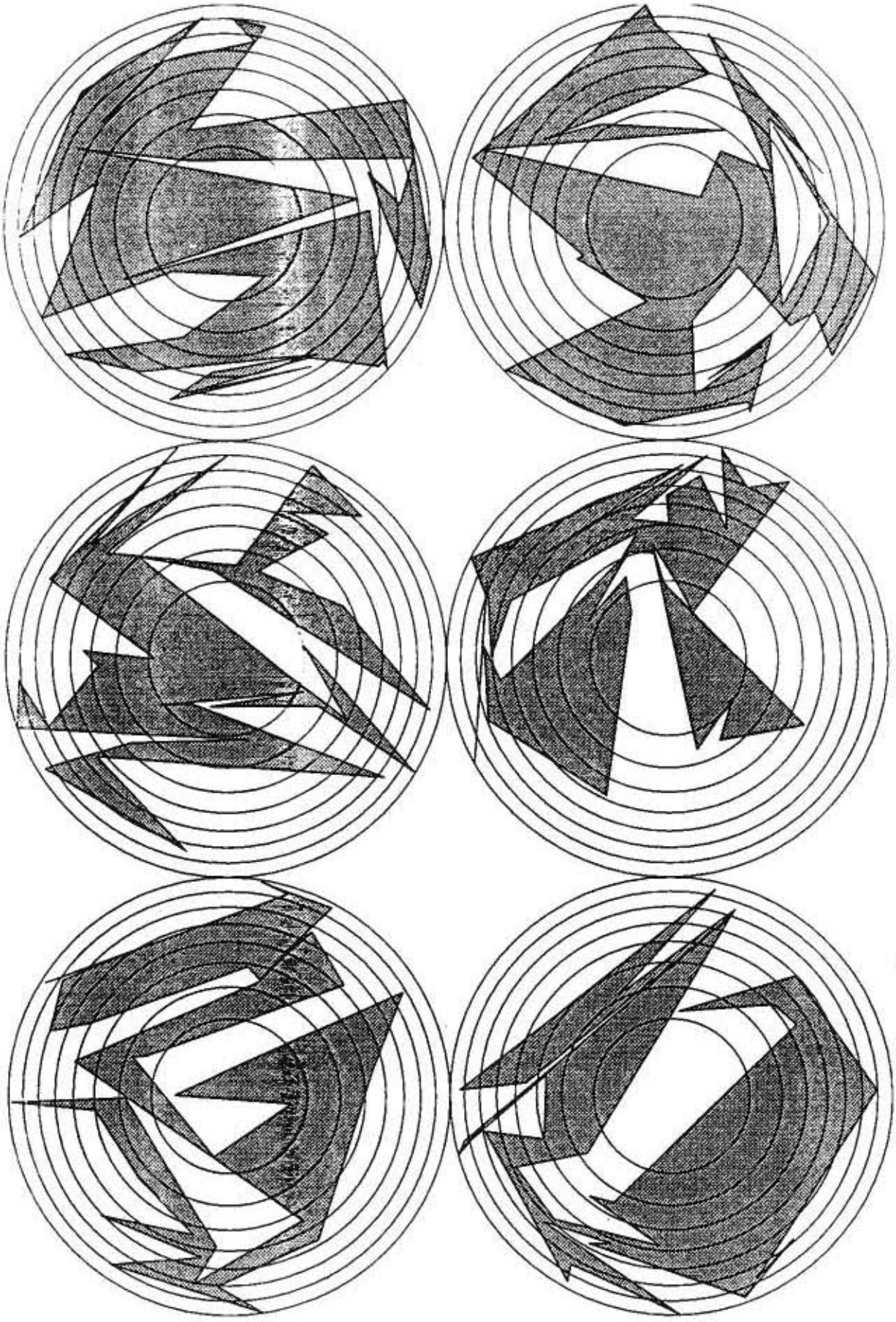

Figure 7: Bottom to top, left to right: $\chi_v = 1.4, 0.3, 0.2, 1.0, 1.2, 1.2$

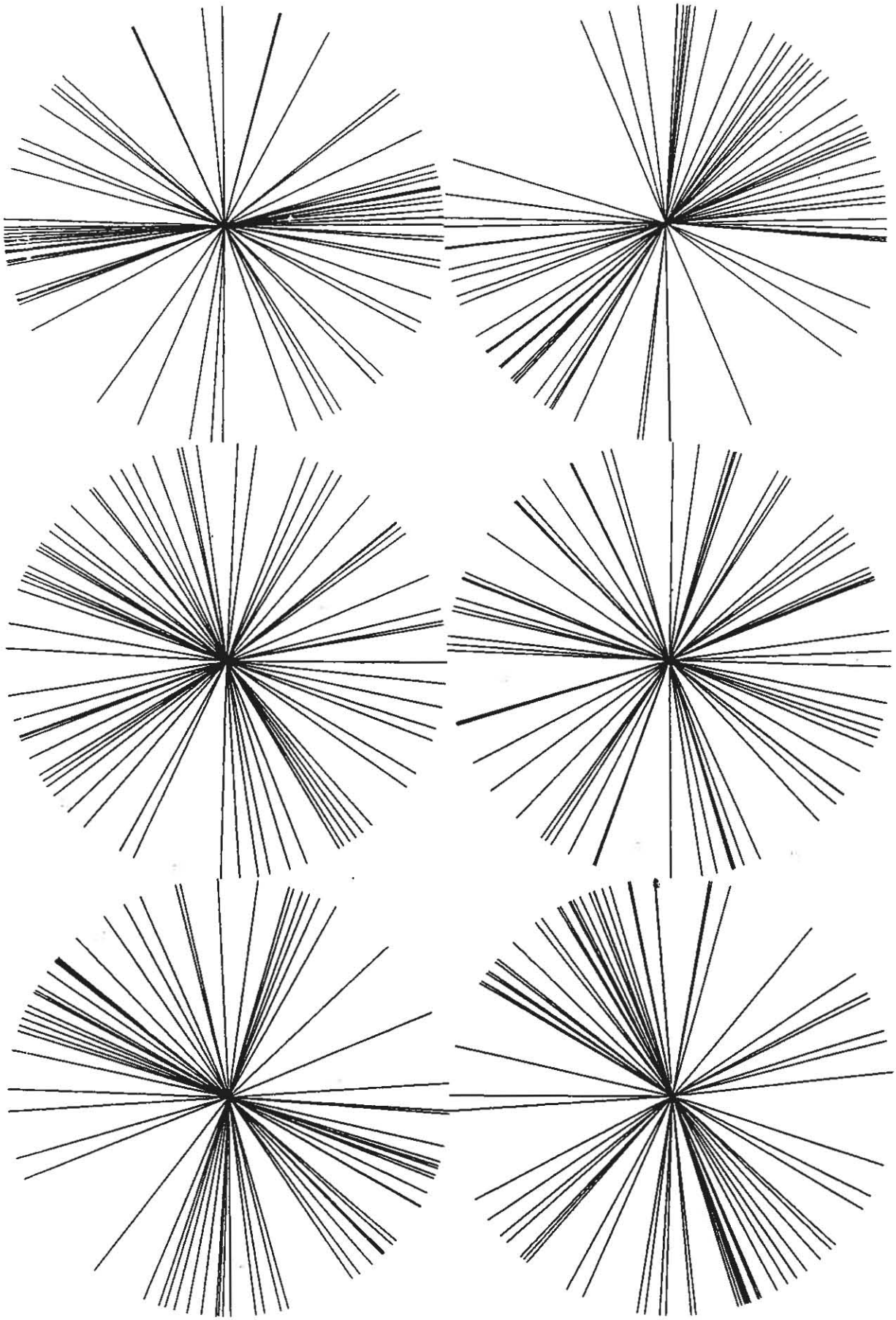

Figure 8: Bottom to top, left to right: $\chi_e = 2.4, 2.7, 0.8, 0.4, 3.9, 3.1$

at the 90% confidence level most of the time, with an average confidence at about the 95% level or better.

7 Discussion

We have no illusion that the two measures we used are adequate to characterize randomness in polygons. It seems that random polygons should have a certain "convolutedness." A precise measure of this suggests itself: define the *convolutedness* of a polygon to be maximum link distance of a path from any point in the plane exterior to the polygon to infinity. Here the path must avoid the interior of the polygon, and its link distance is one plus the number of turns in the path. For example, the polygon in Fig. 14 has convolutedness of 3, because there is a point (shown) that requires three segments (two turns) to reach infinity.

A convex polygon has convolutedness 0, and a spiral polygon has high convolutedness. It seems that for a fixed n, our algorithm should evolve to a convolutedness dependent on n. If the limiting mean value of convolutedness were known, then this would give a method of defining "sufficient mixing": run the algorithm until the expected convolutedness is attained.

We therefore pose as an open problem: what is the expected convolutedness as a function of n for our algorithm as $t \to \infty$?

References

[EKM+90] P. Epstein, A. Knight, J. May, T. Nguyen, and J.-R. Sack. A workbench for computational geometry (WOCG). Technical report, Carelton University, 1990.

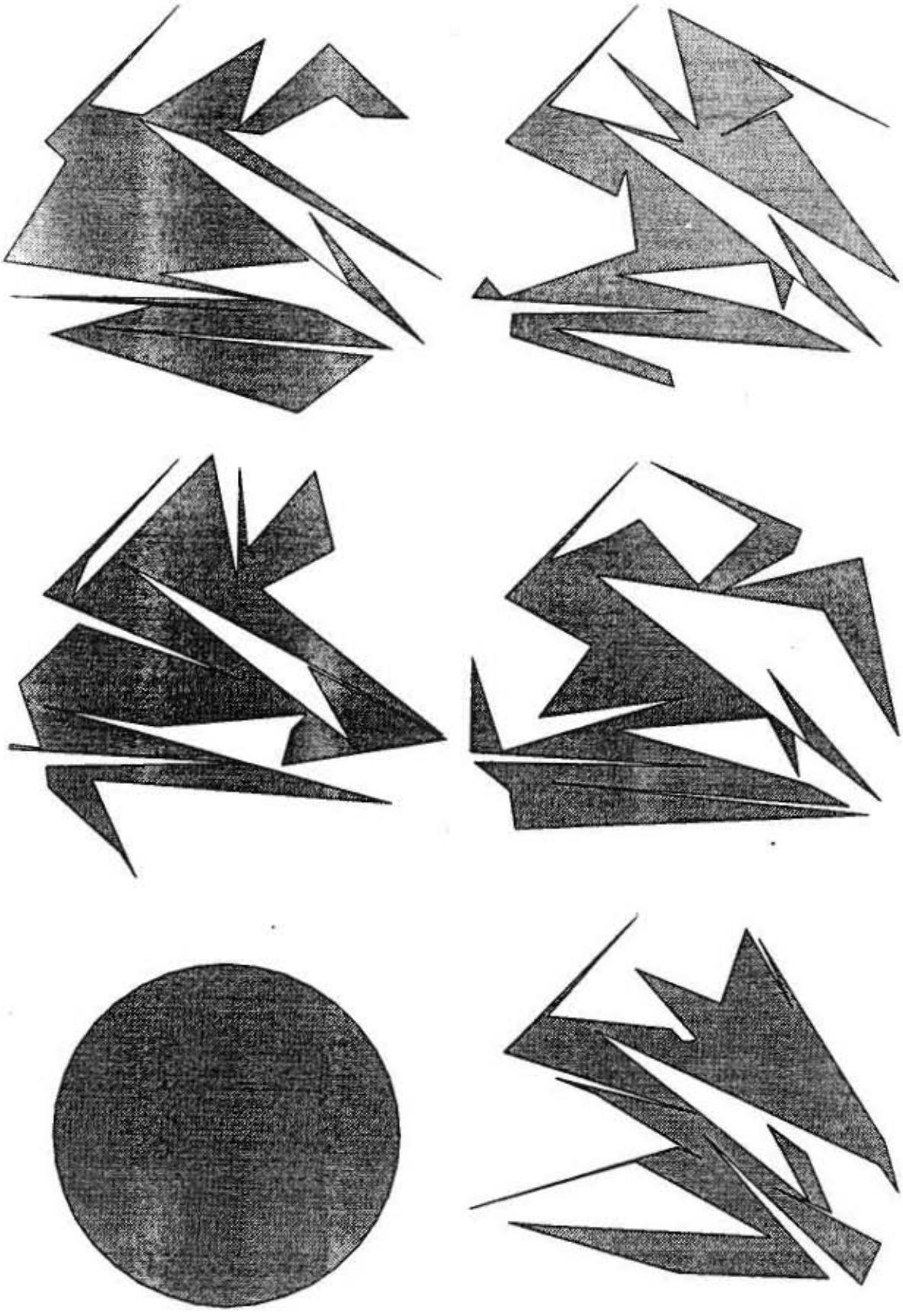

Figure 9: $n = 36$, six snapshots up to $t = 2000$.

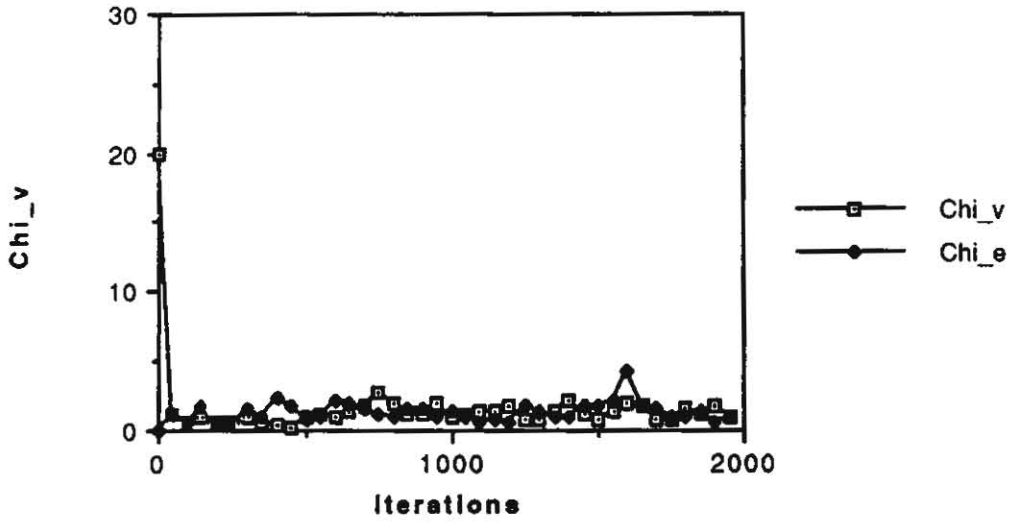

Figure 10: $n = 36$, Chi-square values.

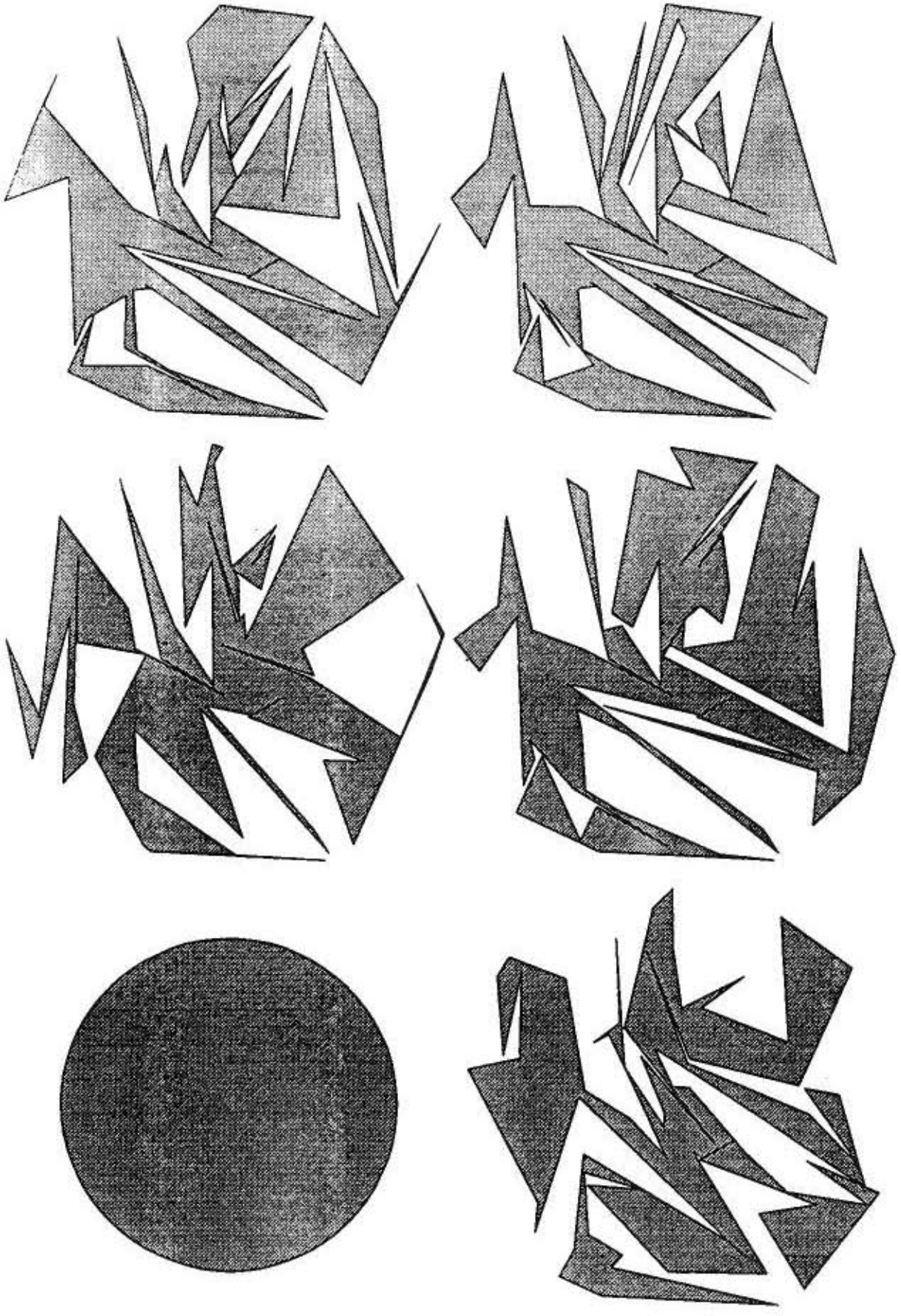

Figure 11: $n = 72$, six snapshots up to $t = 2000$.

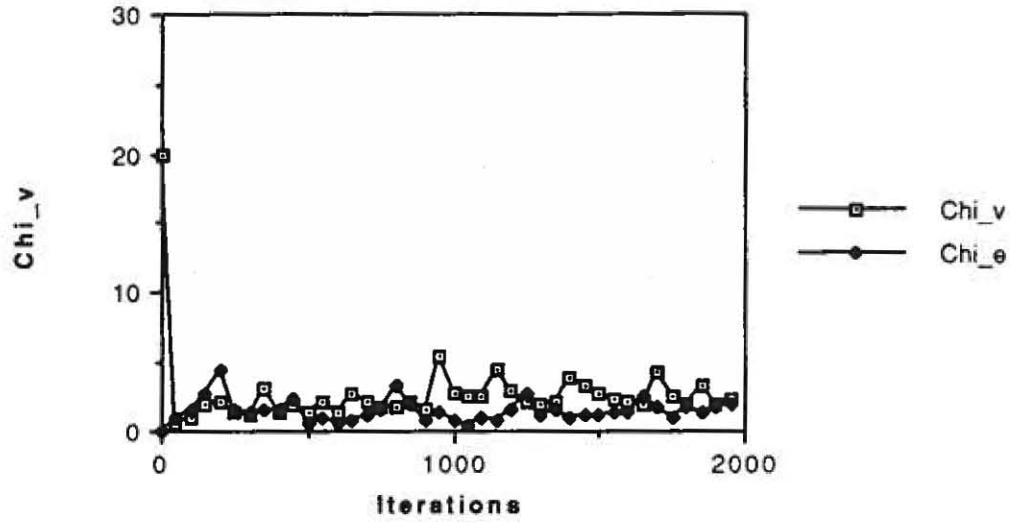

Figure 12: $n = 72$, Chi-square values.

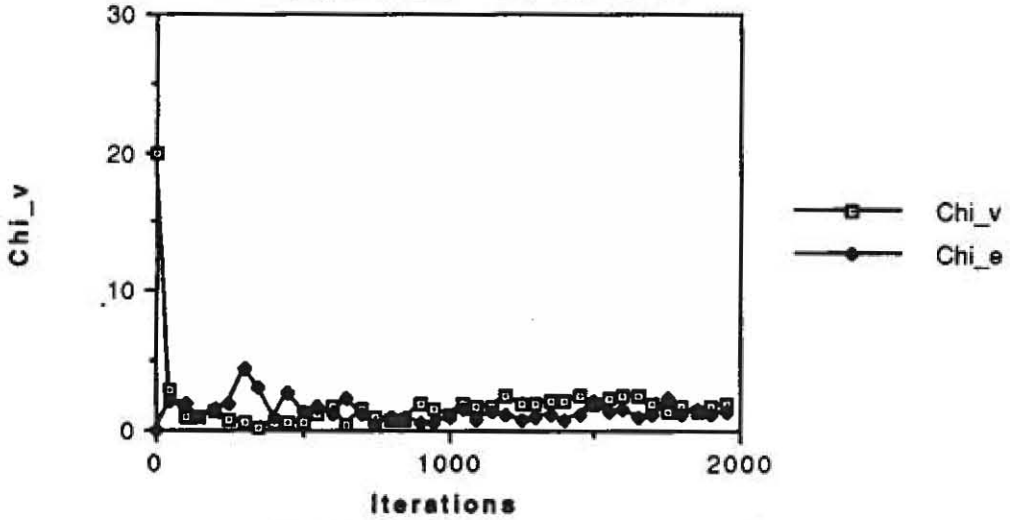

Figure 13: $n = 180$, Chi-square values.

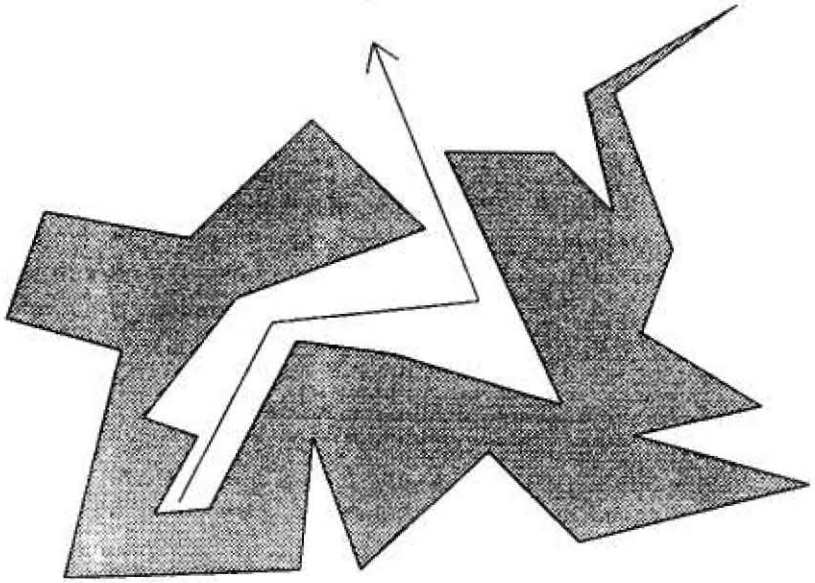

Figure 14: A polygon of convolutedness 3.

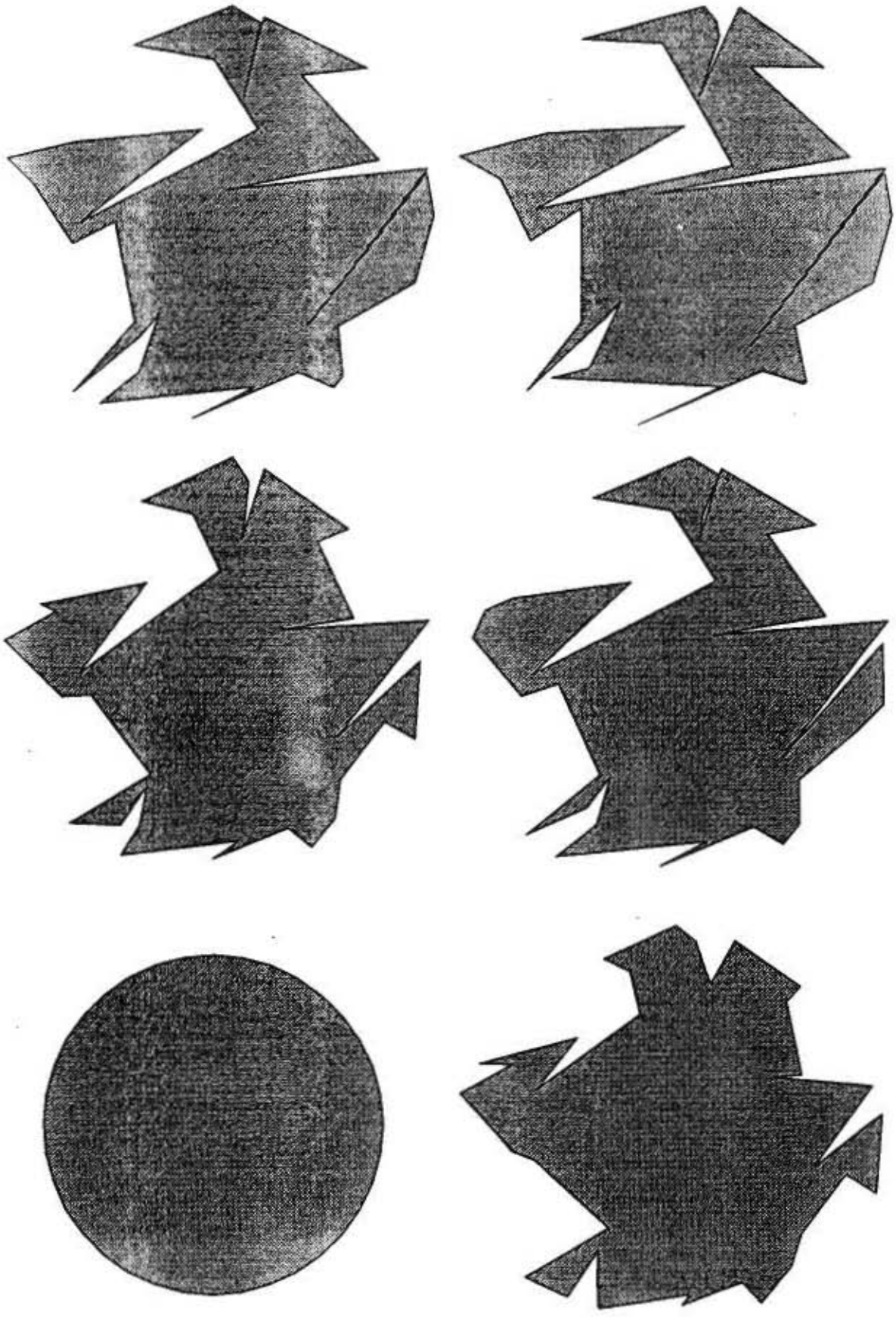

Figure 1: $n = 36$, five time steps (seed = 4).

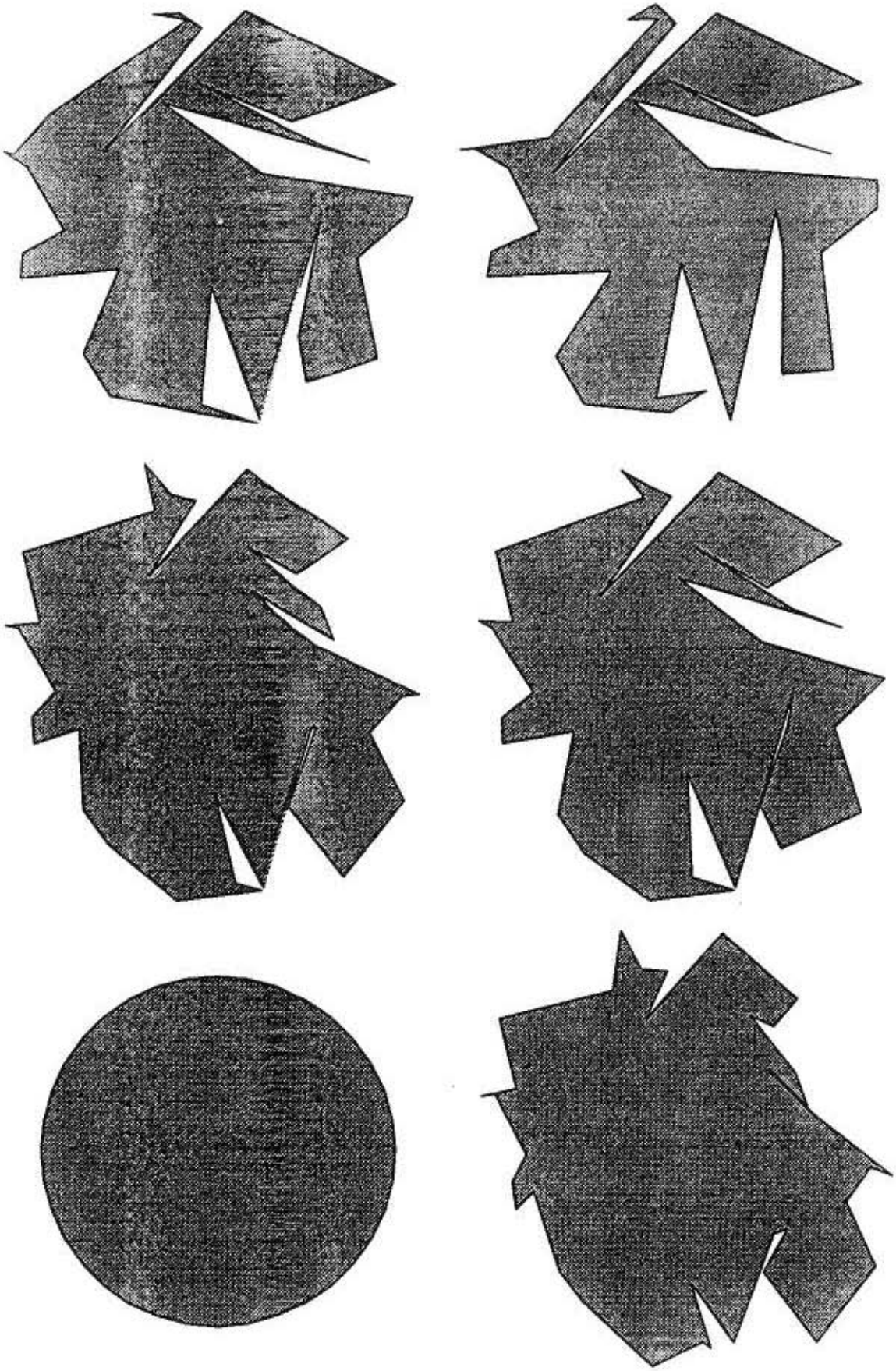

Figure 2: $n = 36$, five time steps (seed = 10).

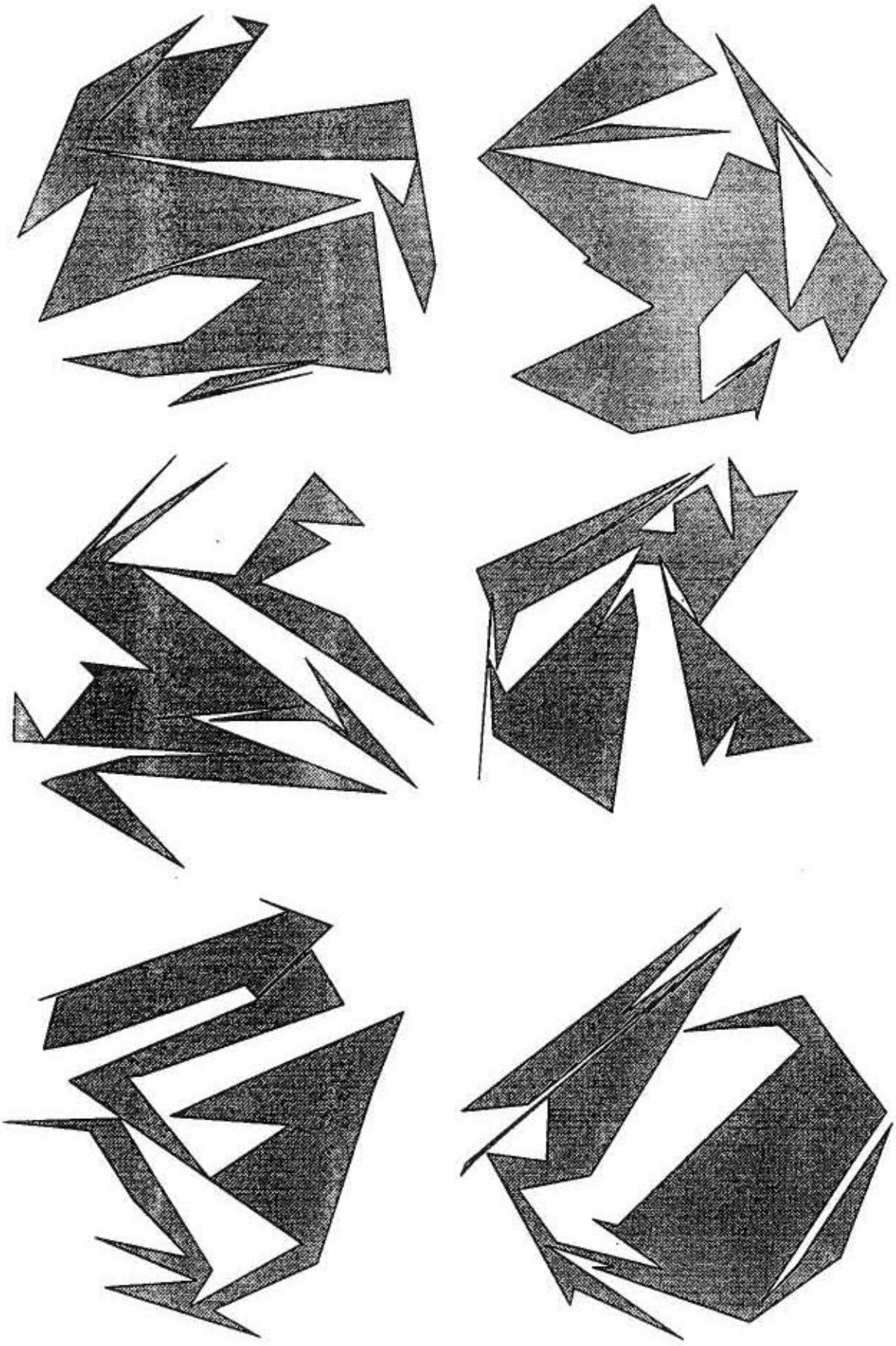

Figure 3: $n = 36$, $t = 1000$, six different random seeds.

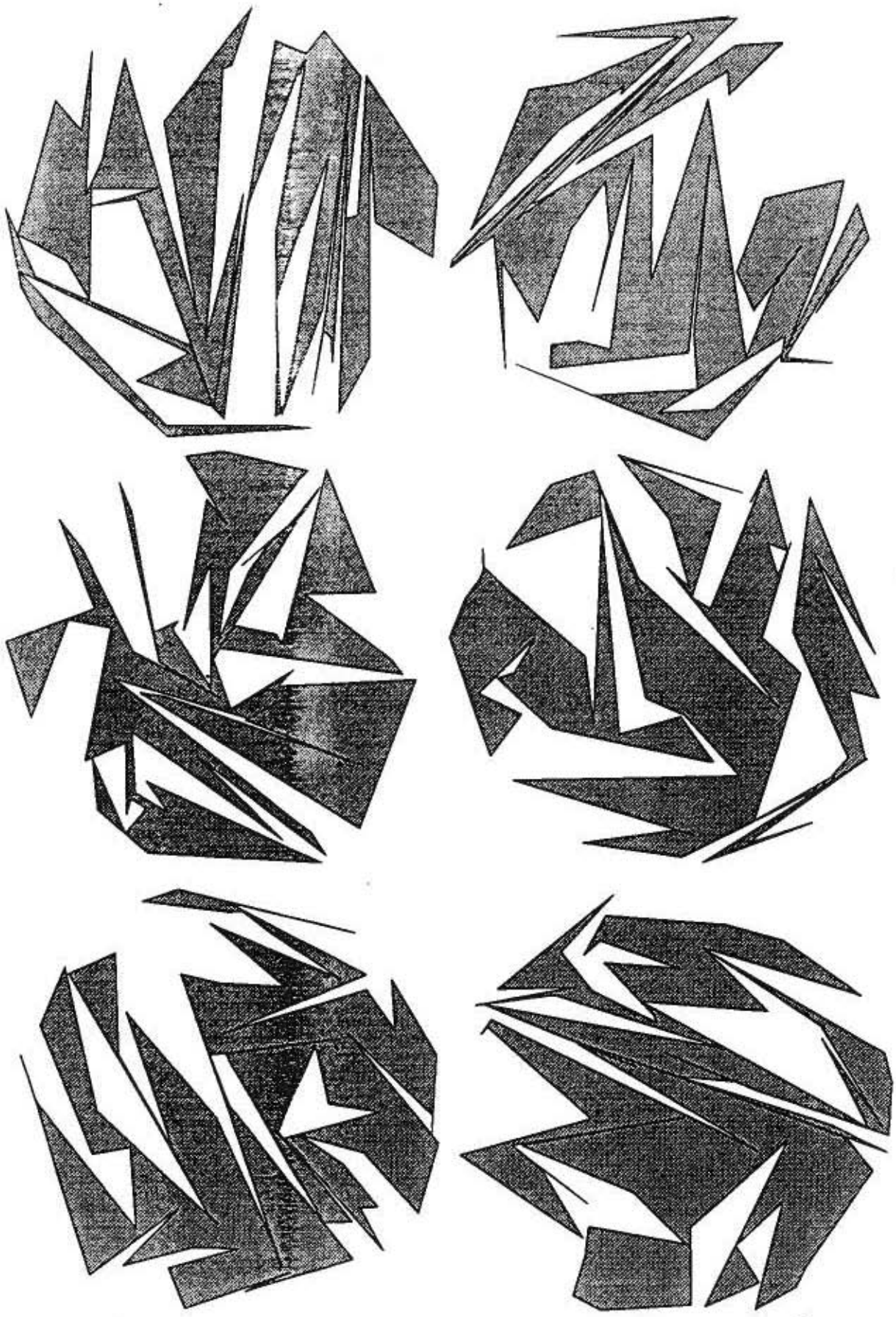

Figure 4: $n = 72$, $t = 1000$, six different random seeds.

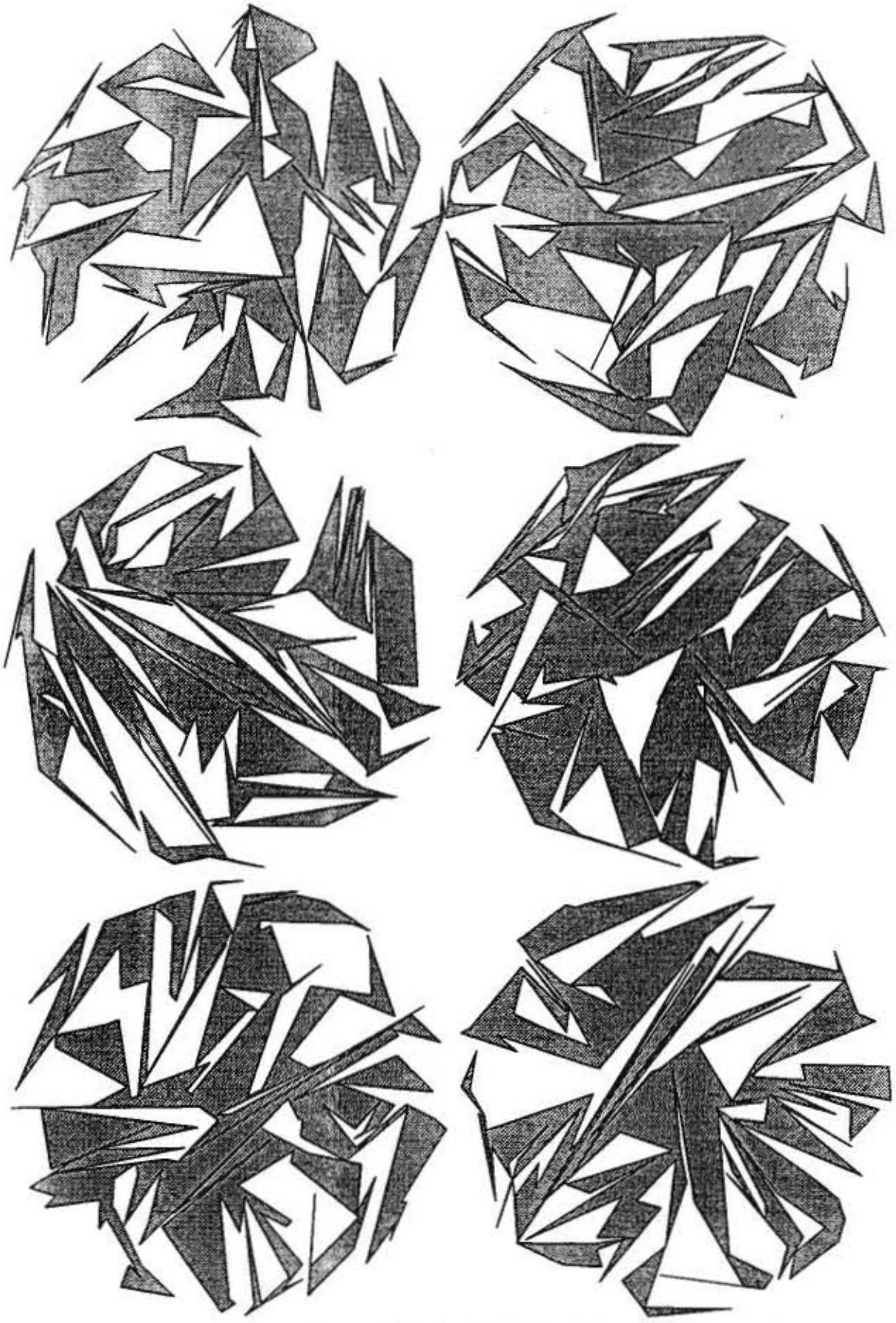

Figure 5: $n = 180$, $t = 1000$, six different random seeds.