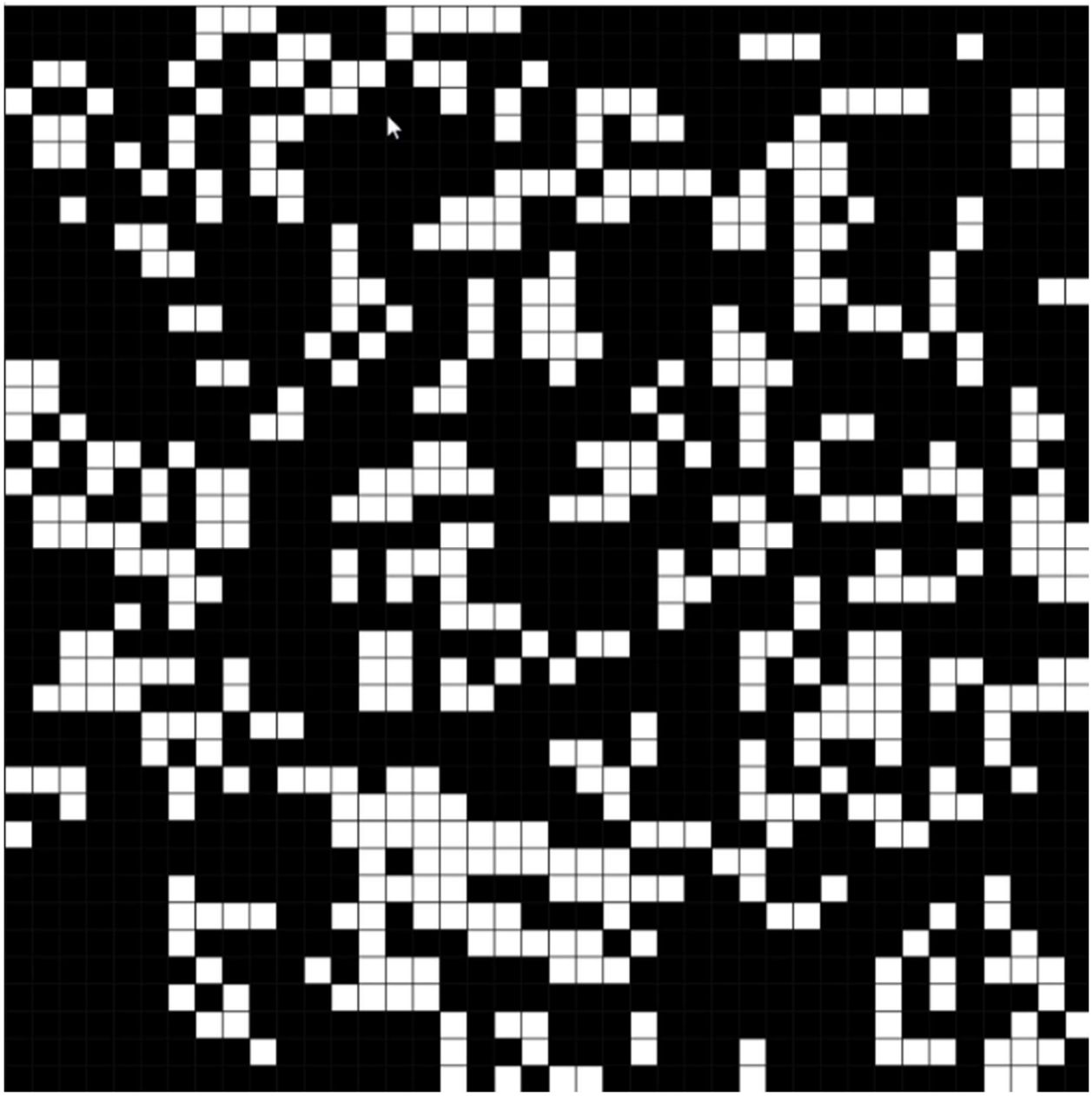
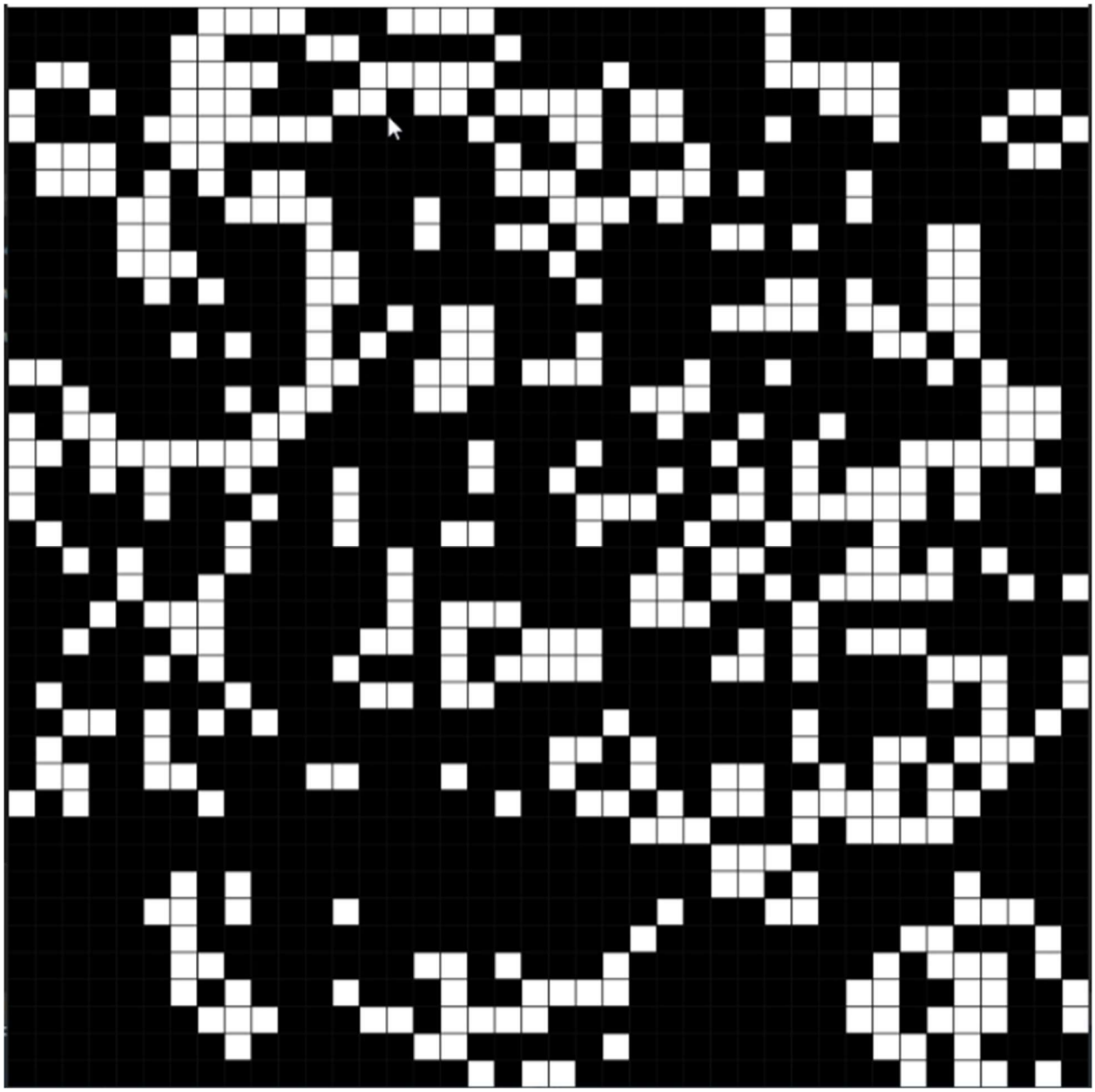
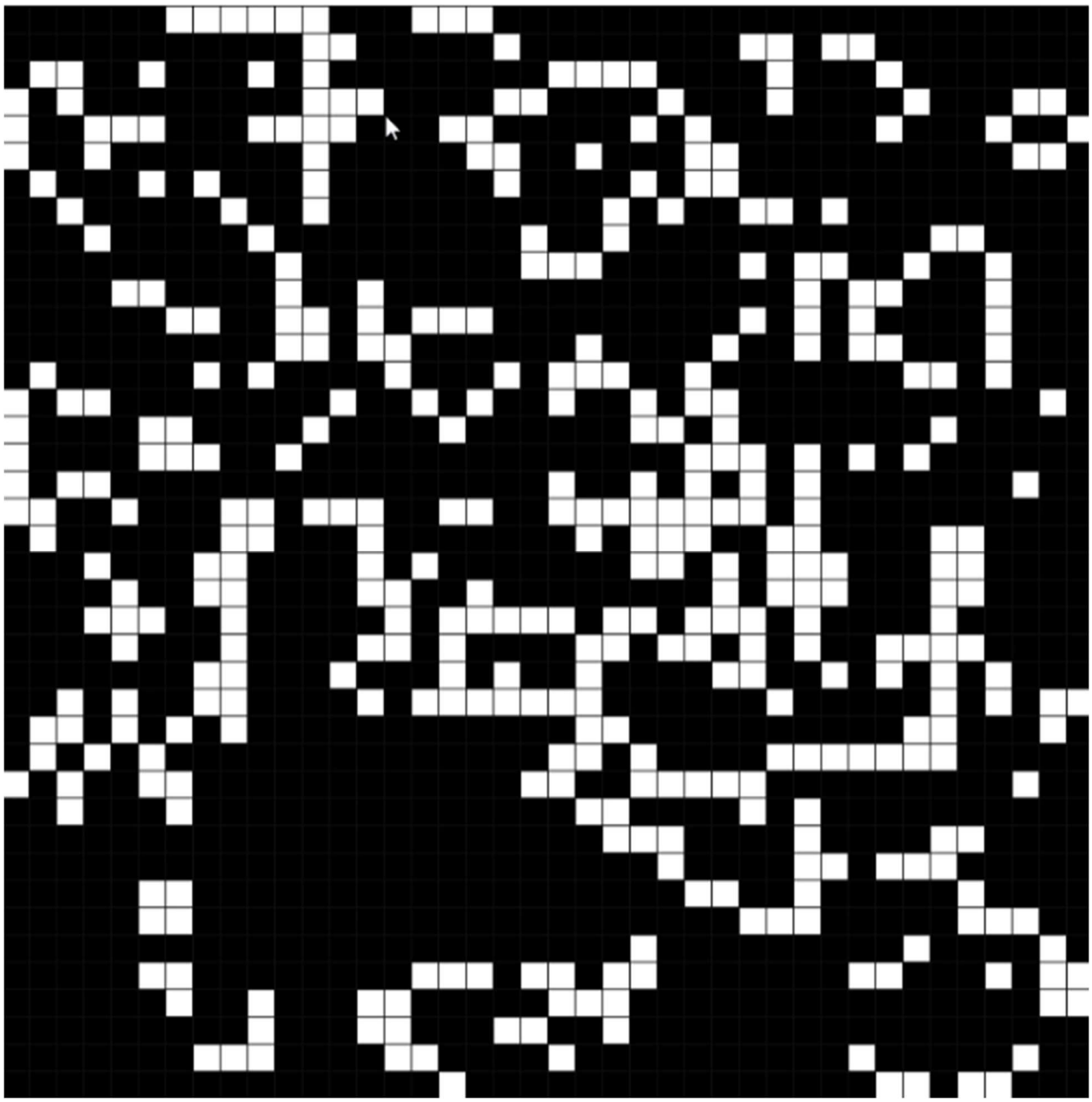
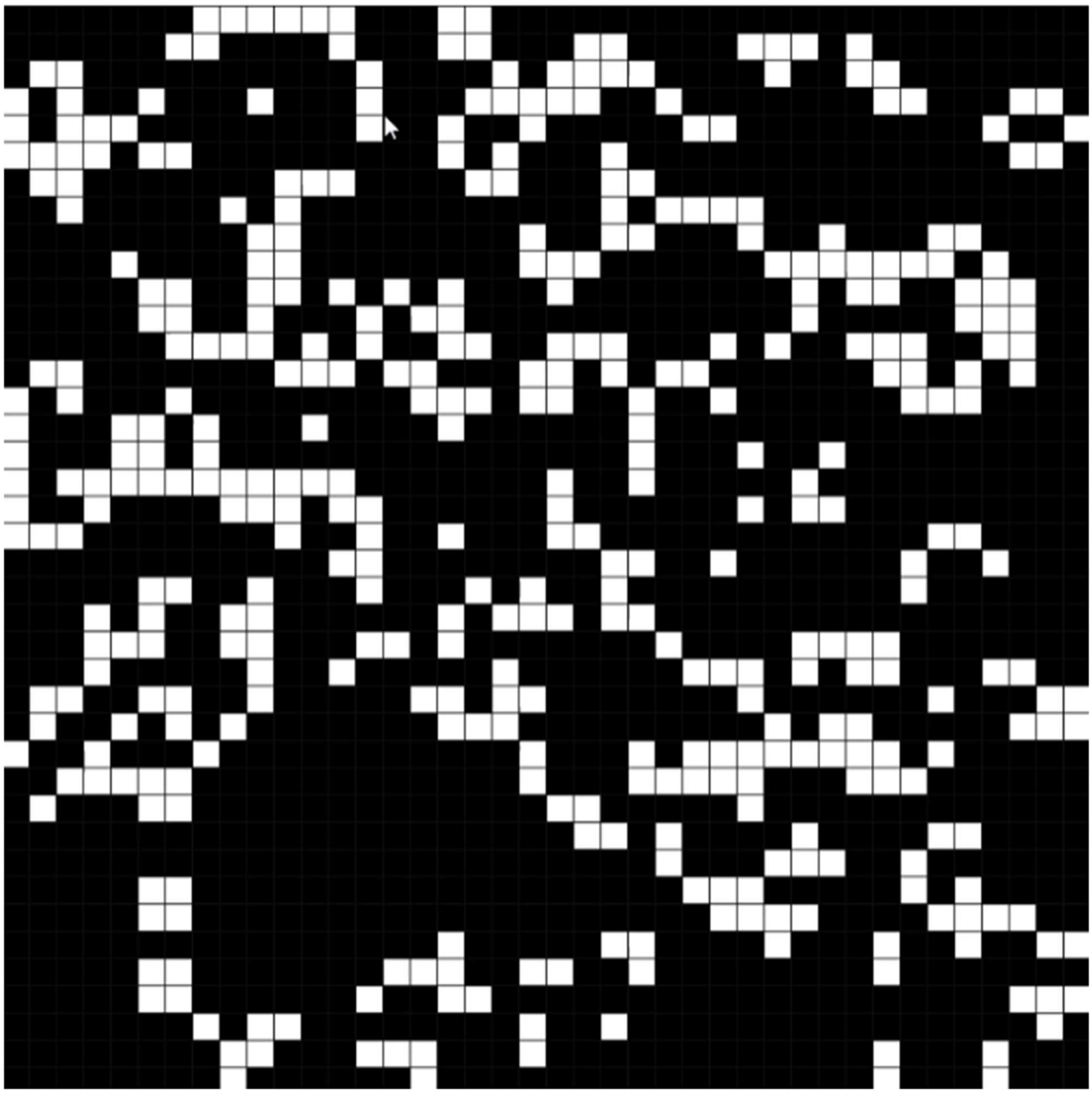
Cagubcub, John Romaine C.
Orticio, Angelo Miguel R.
BCS41

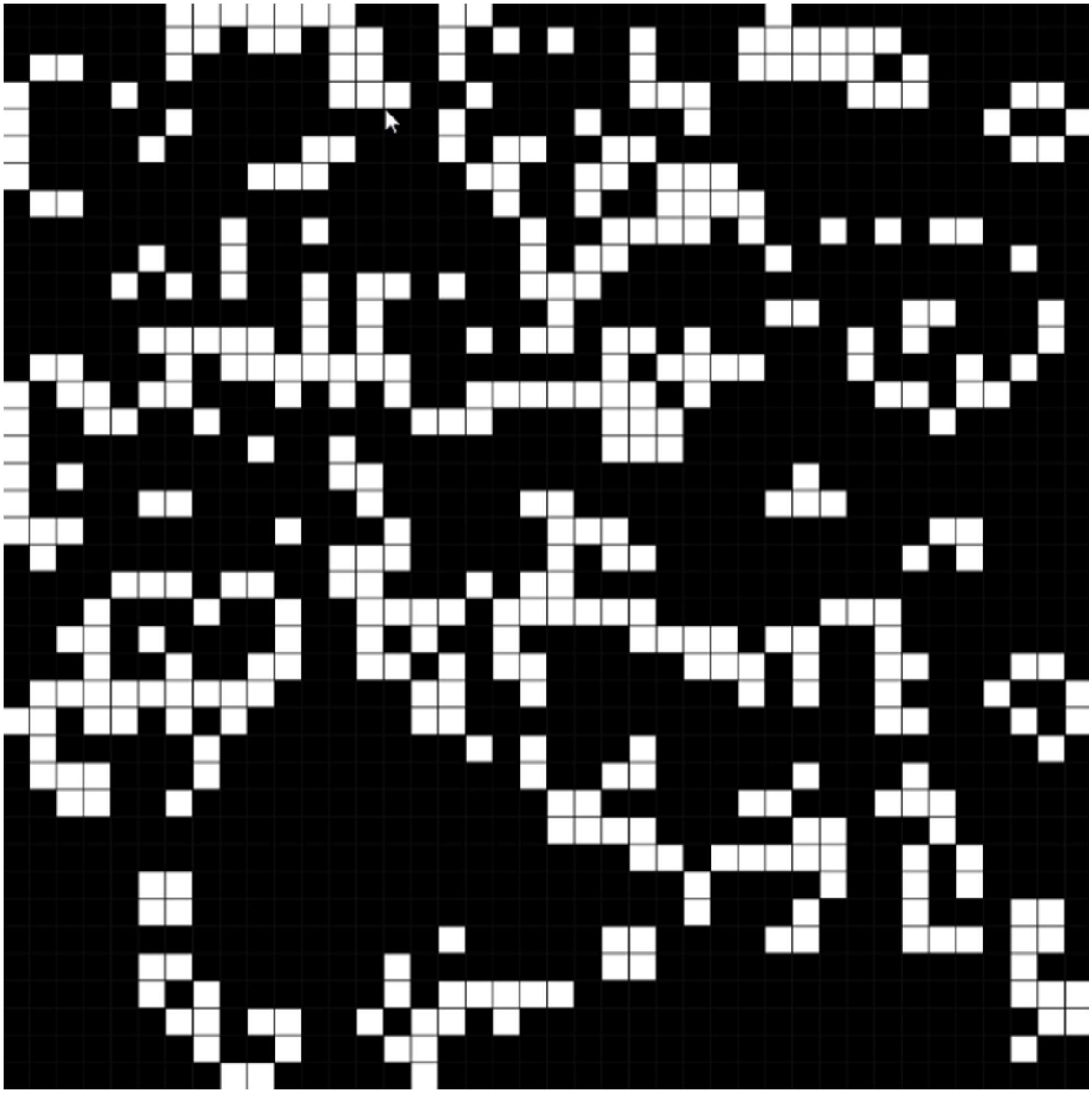
Generations:

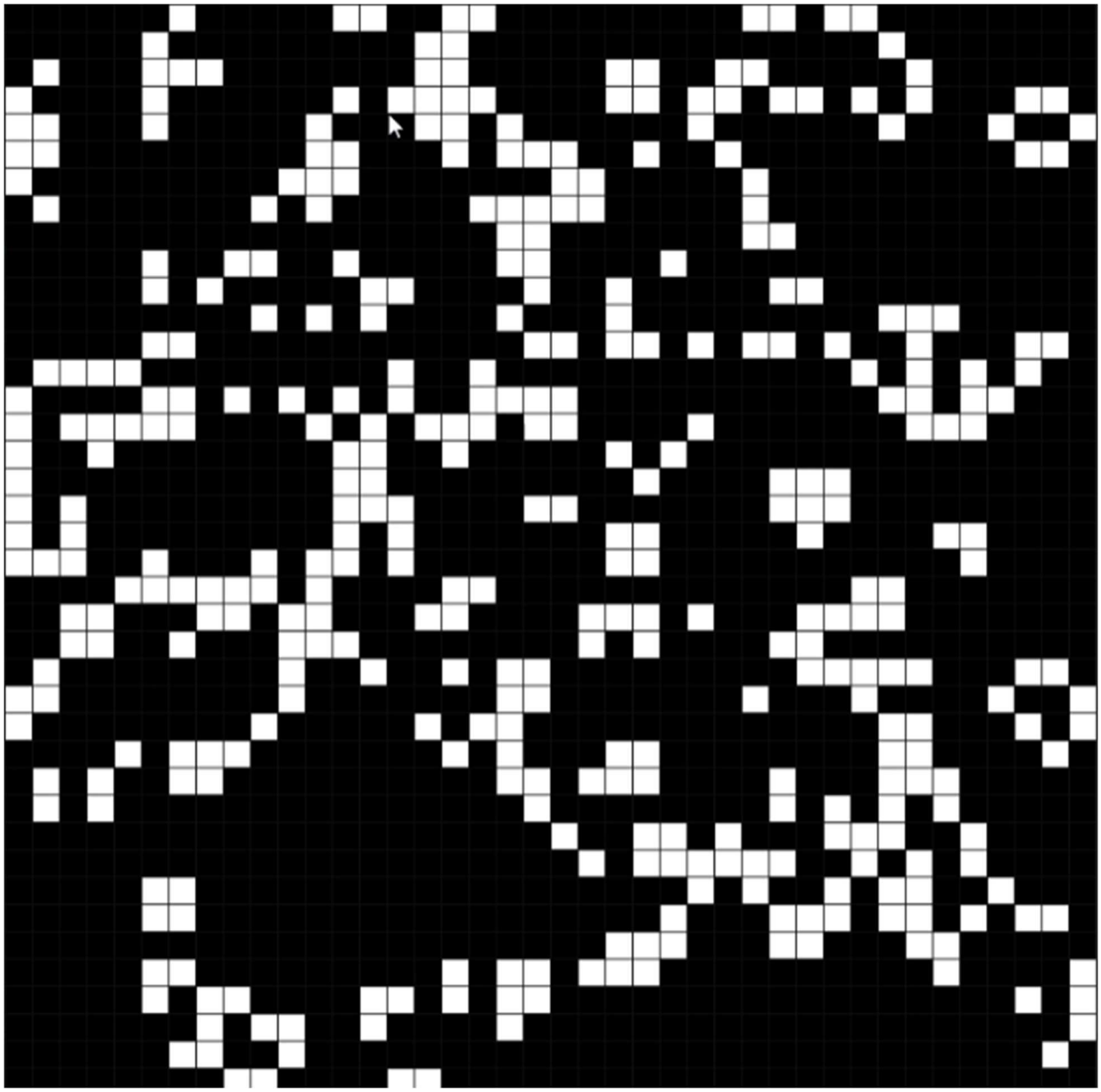


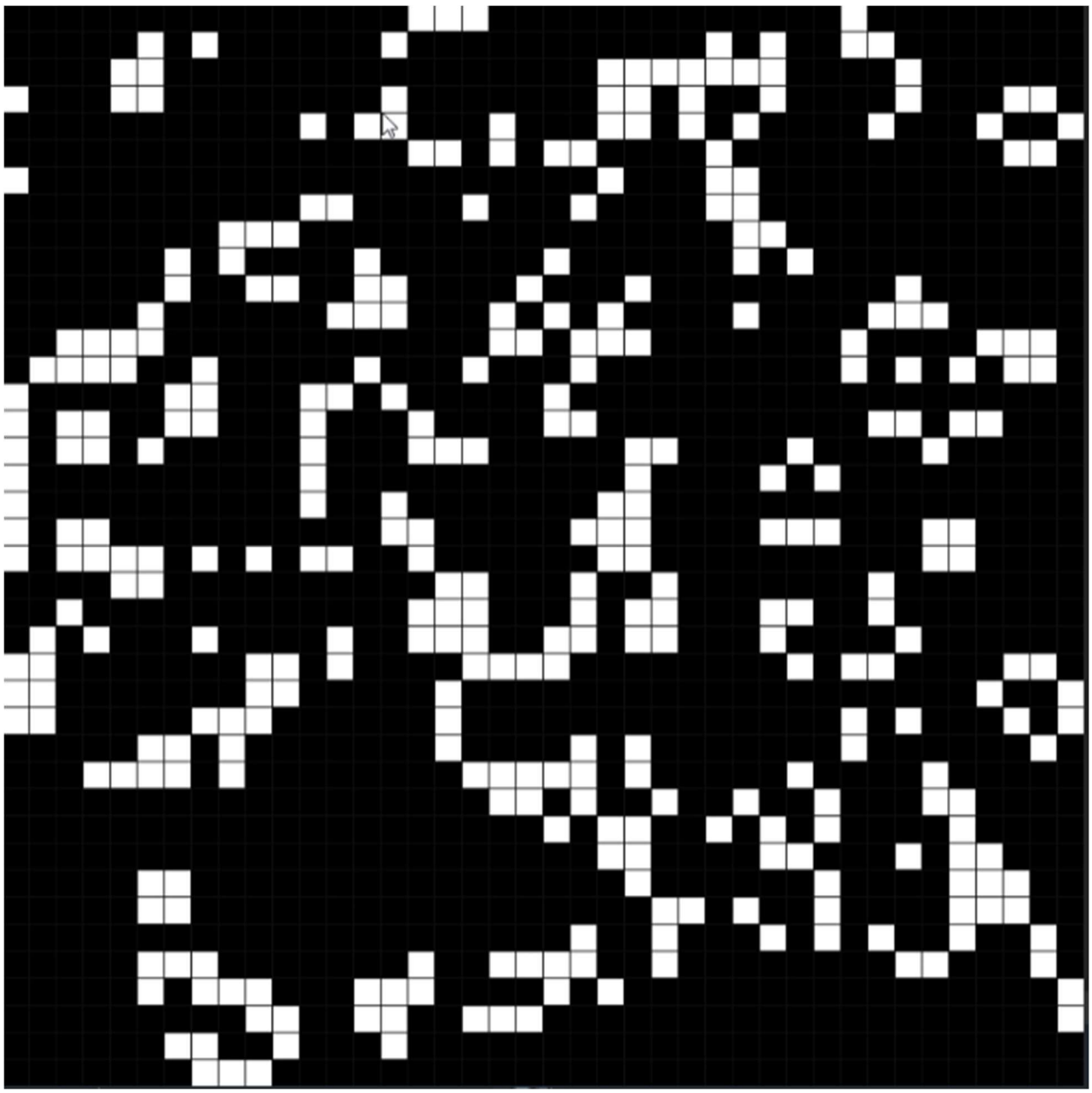


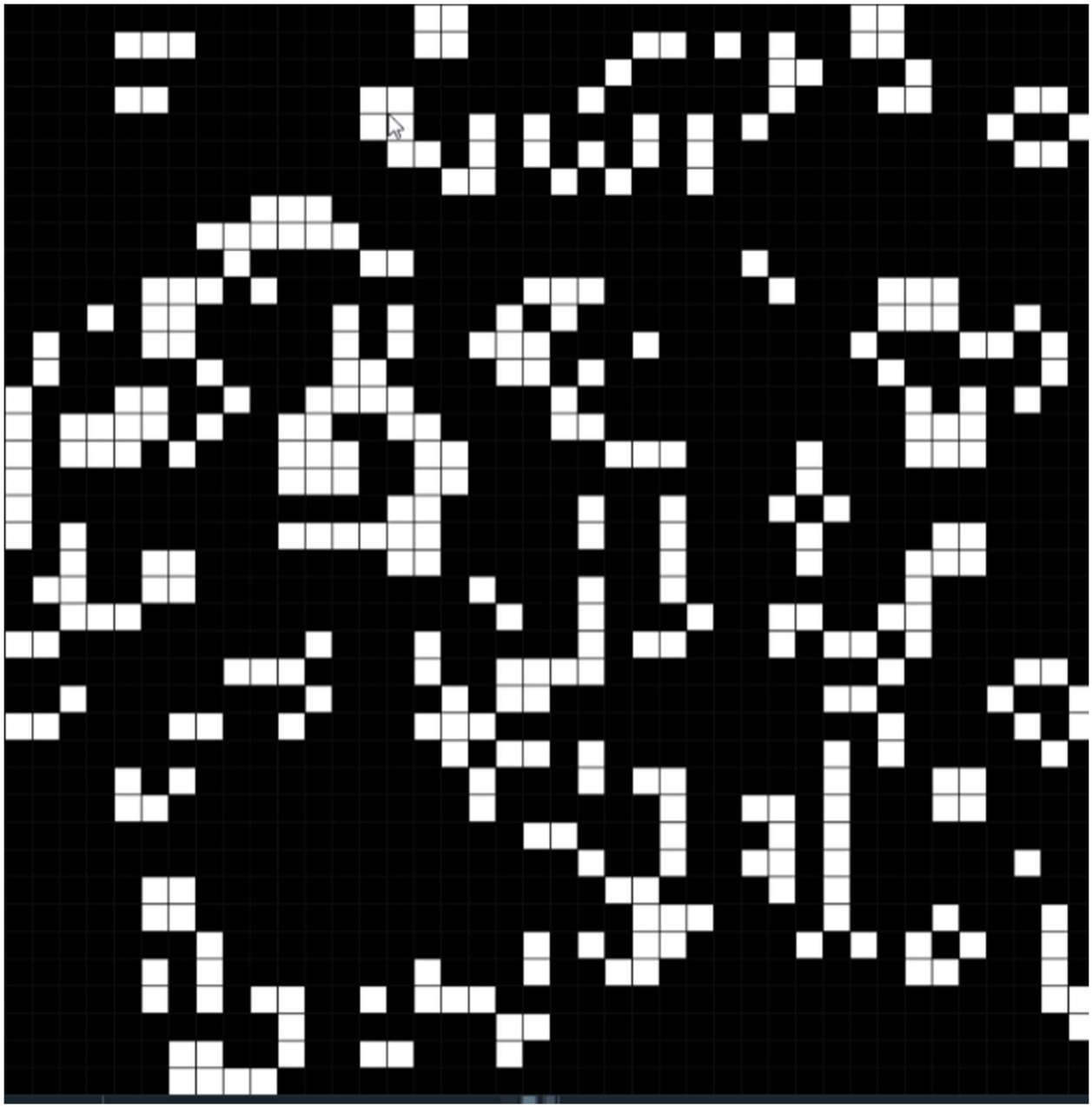


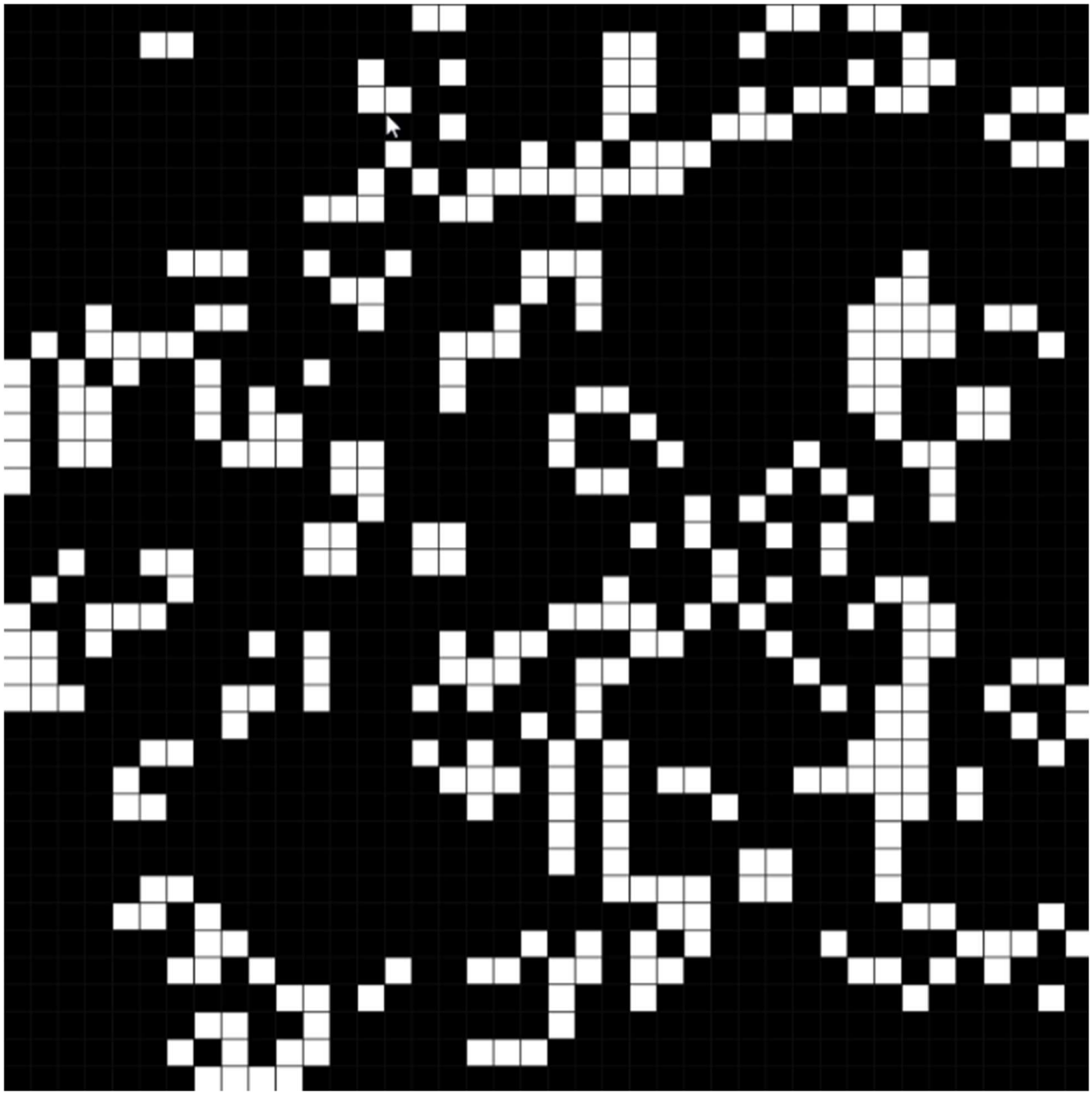


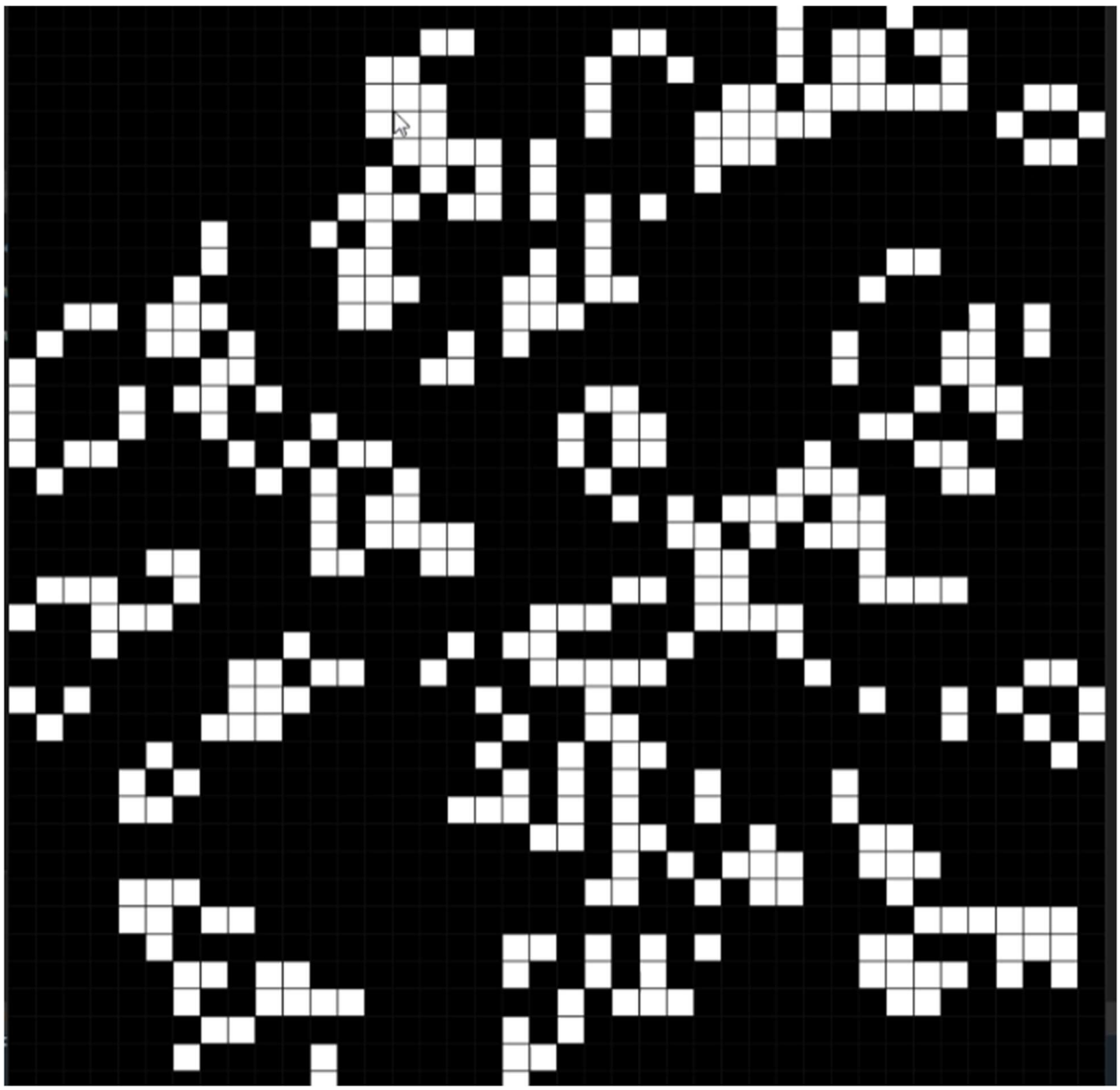


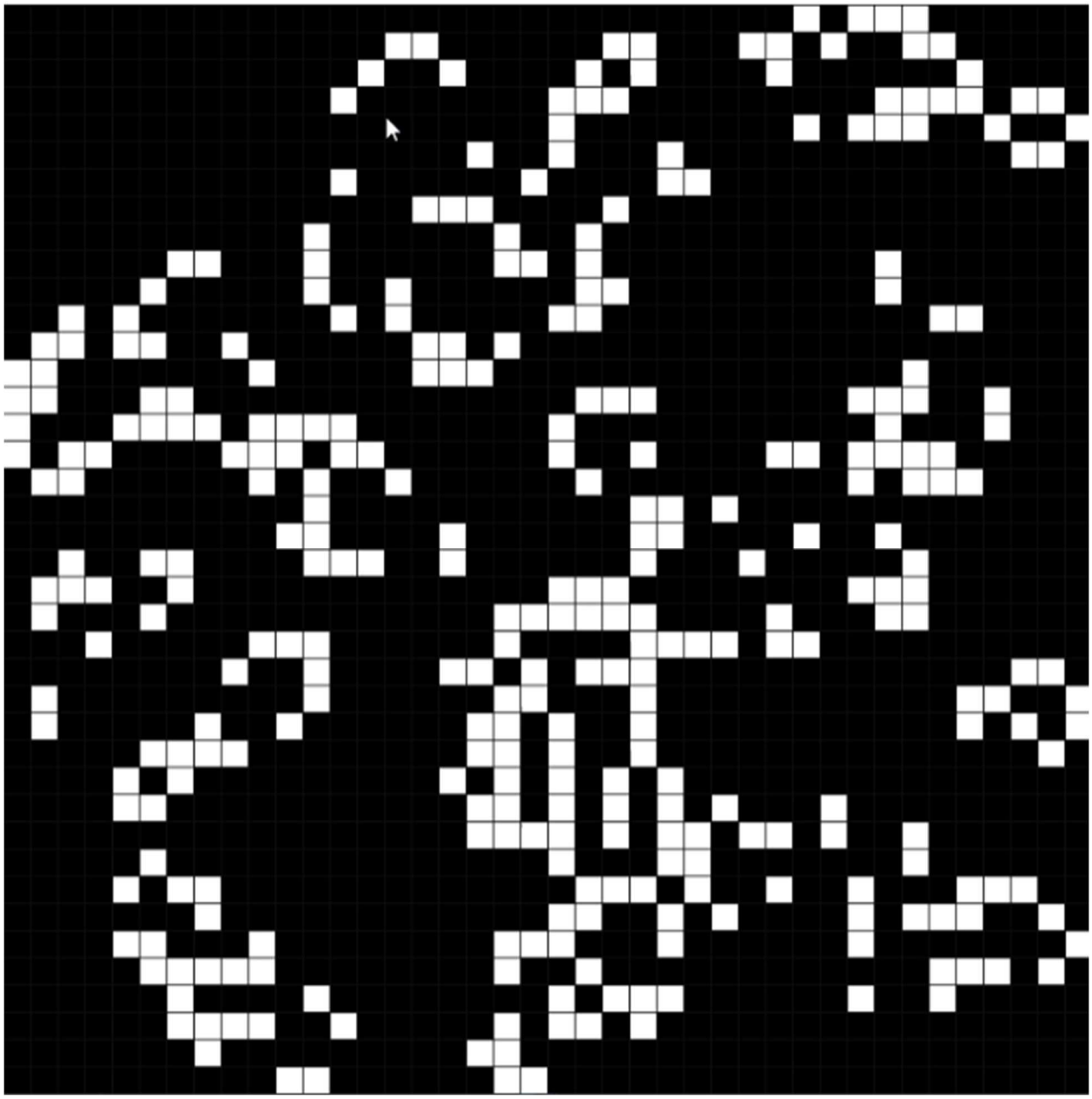


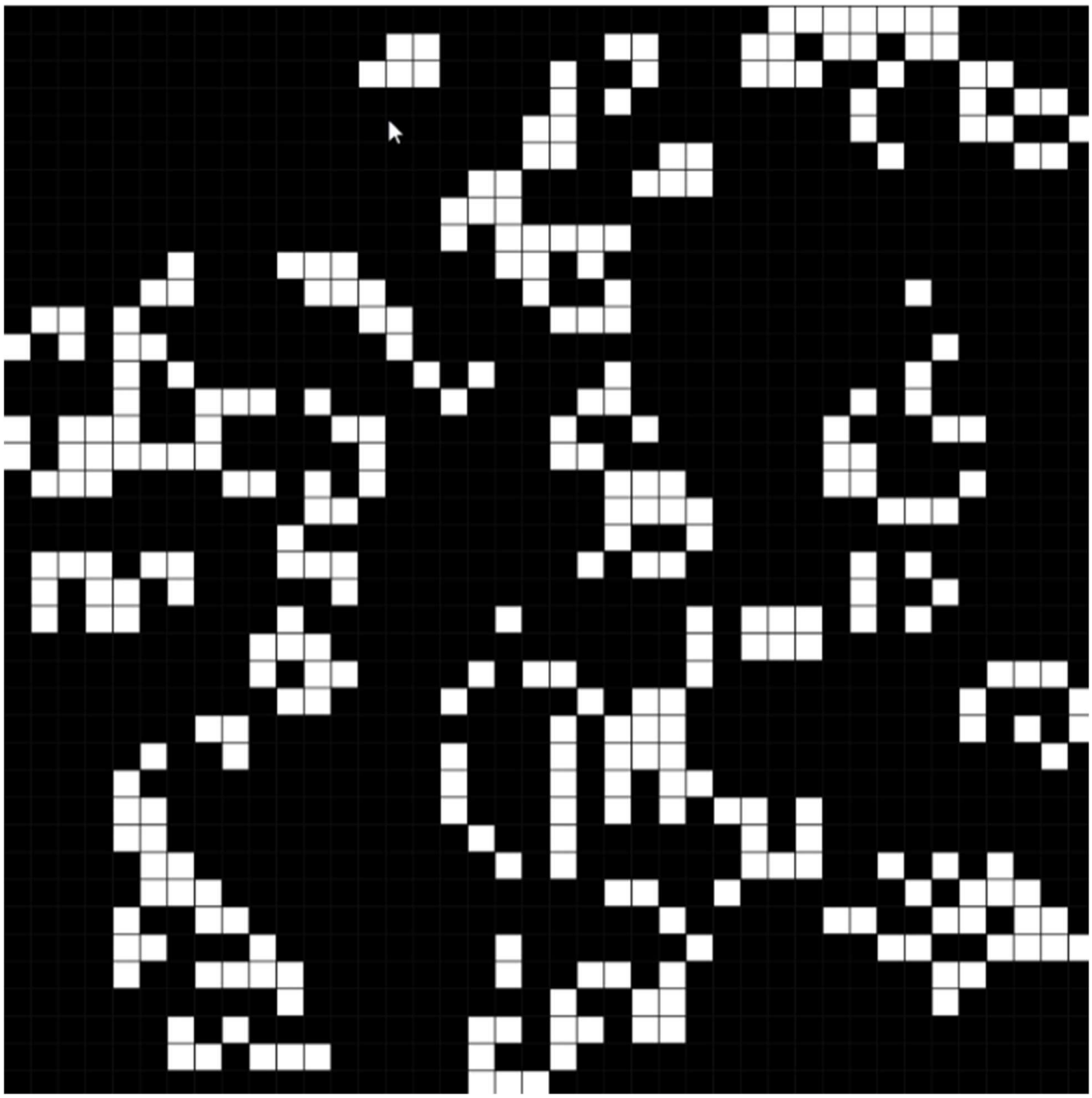


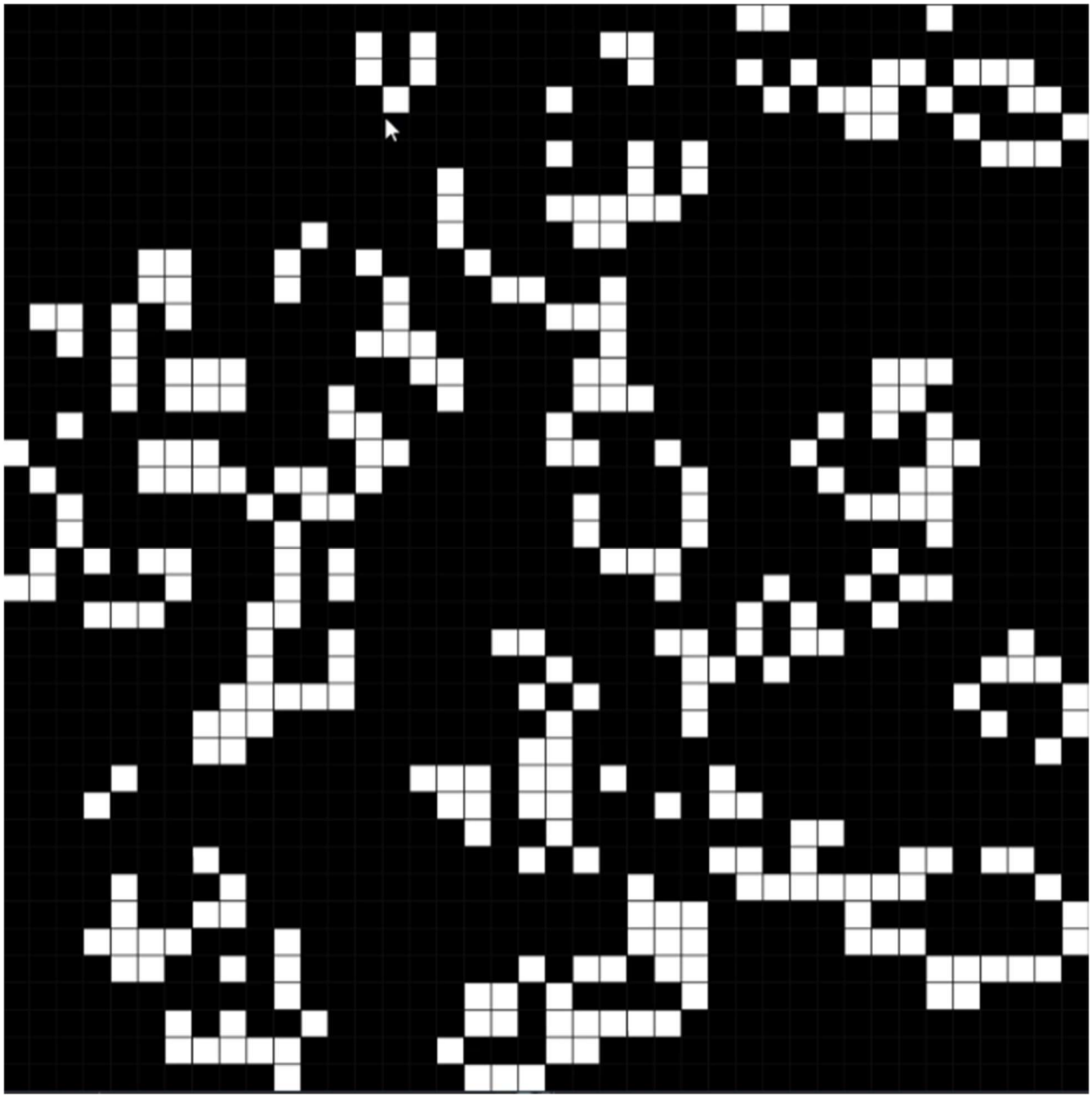


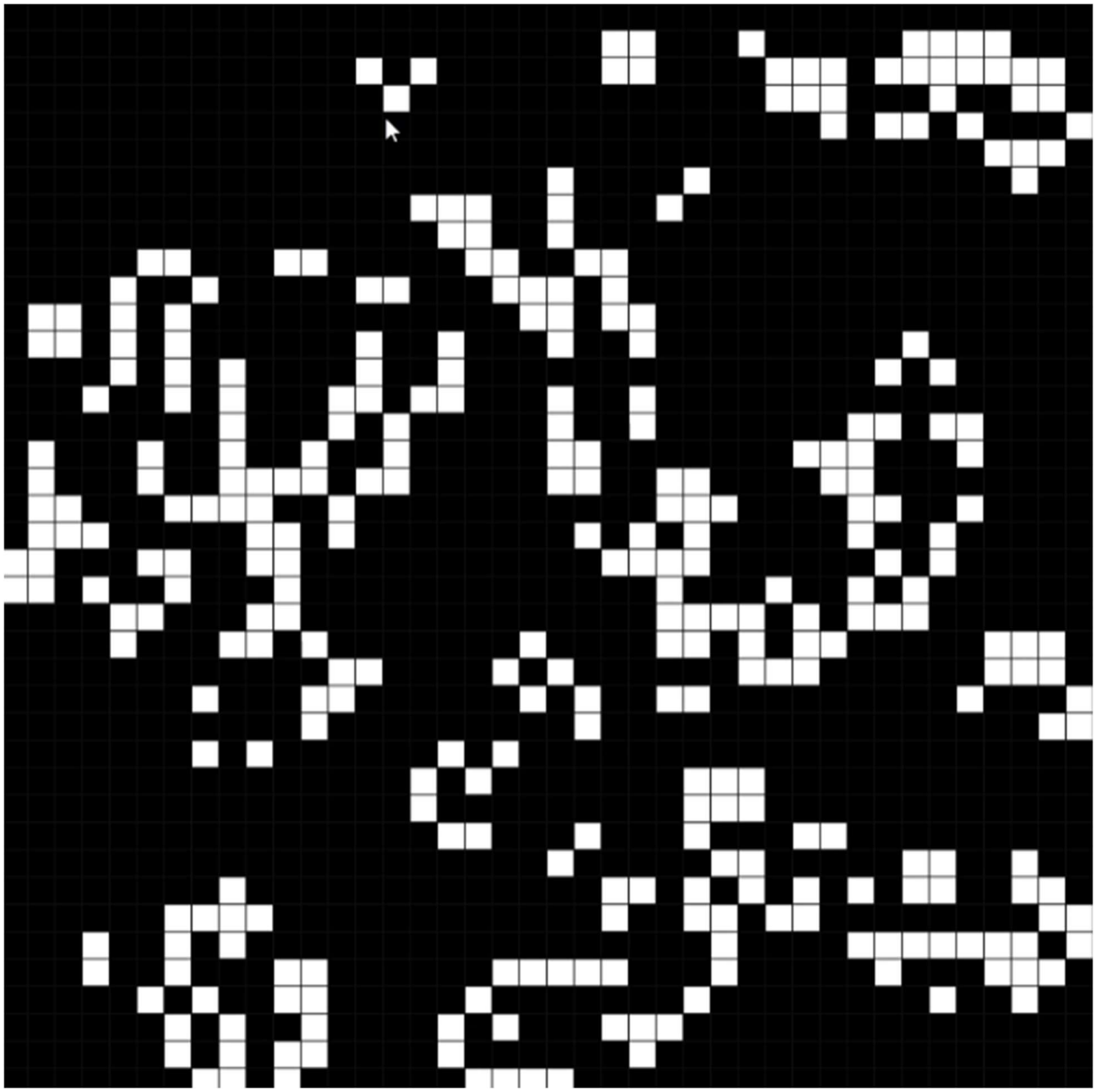


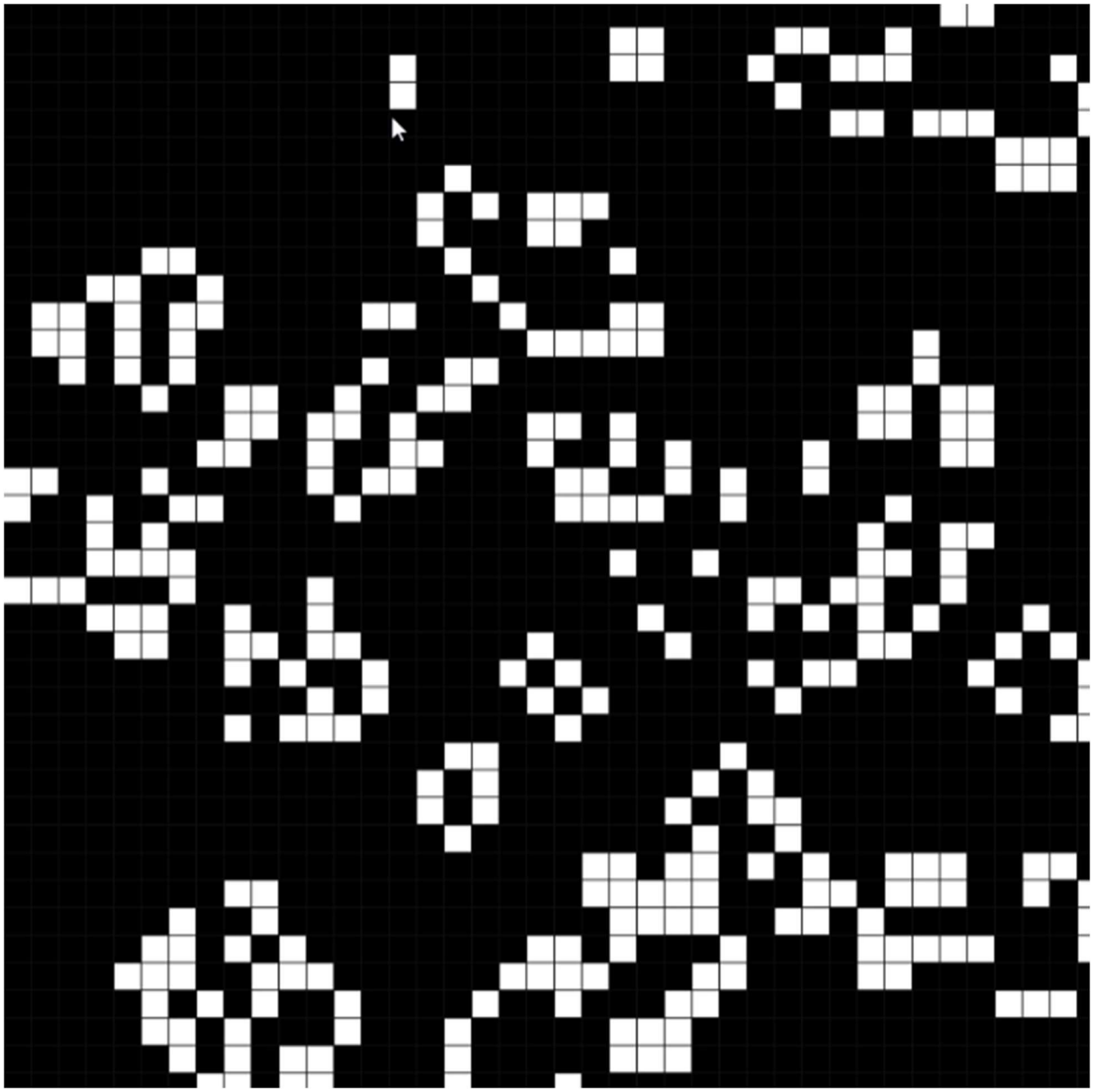


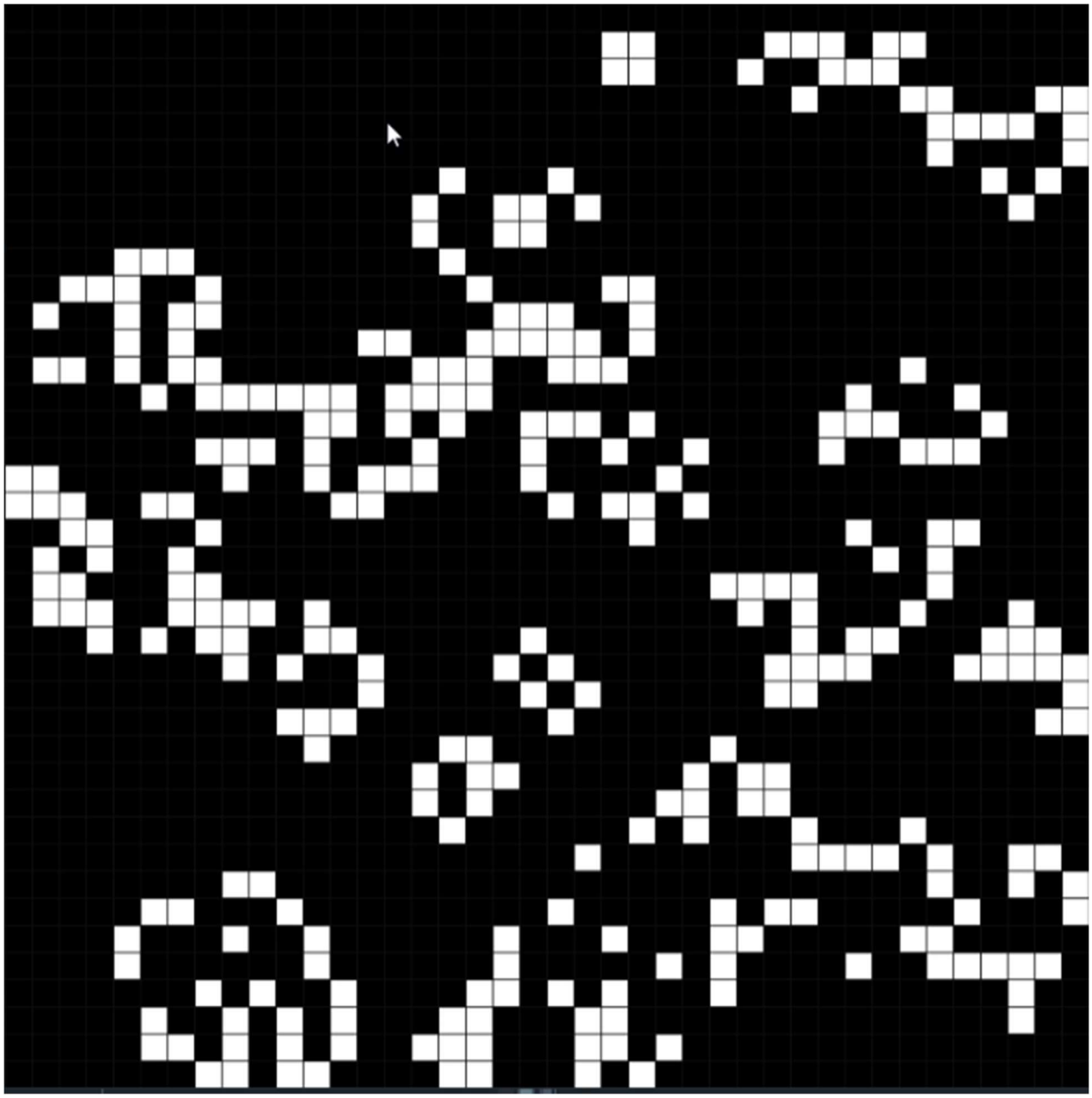


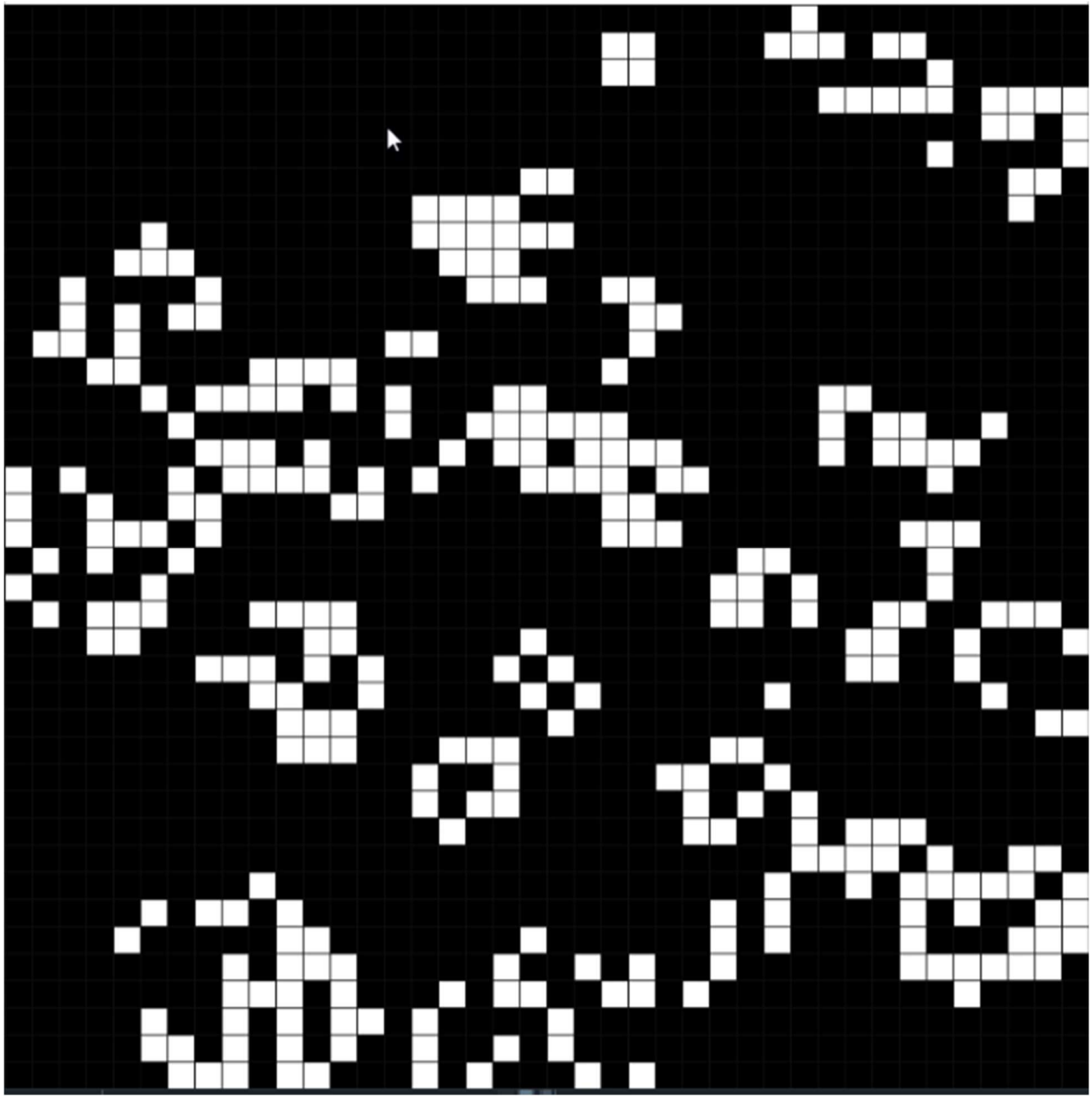


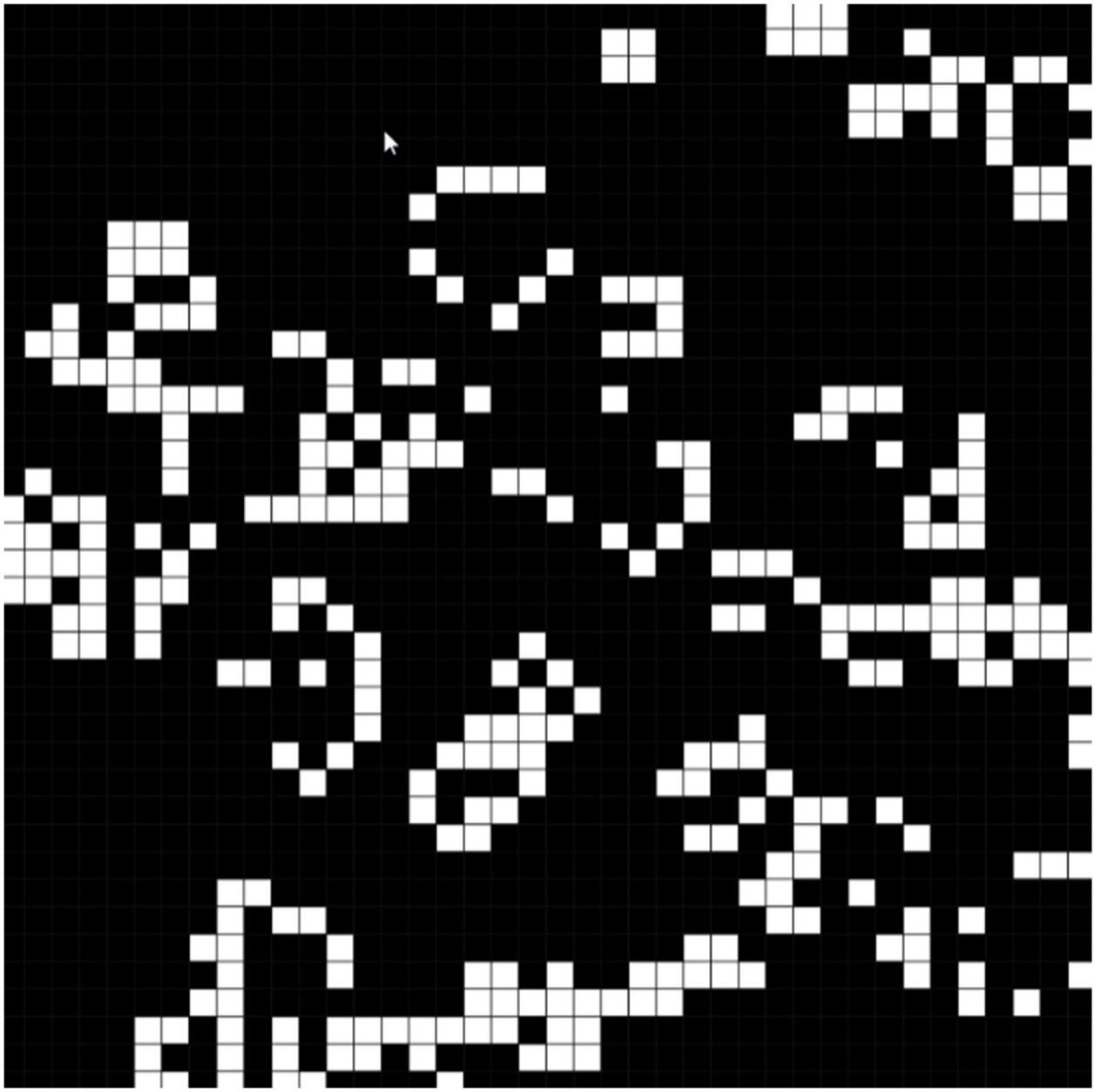


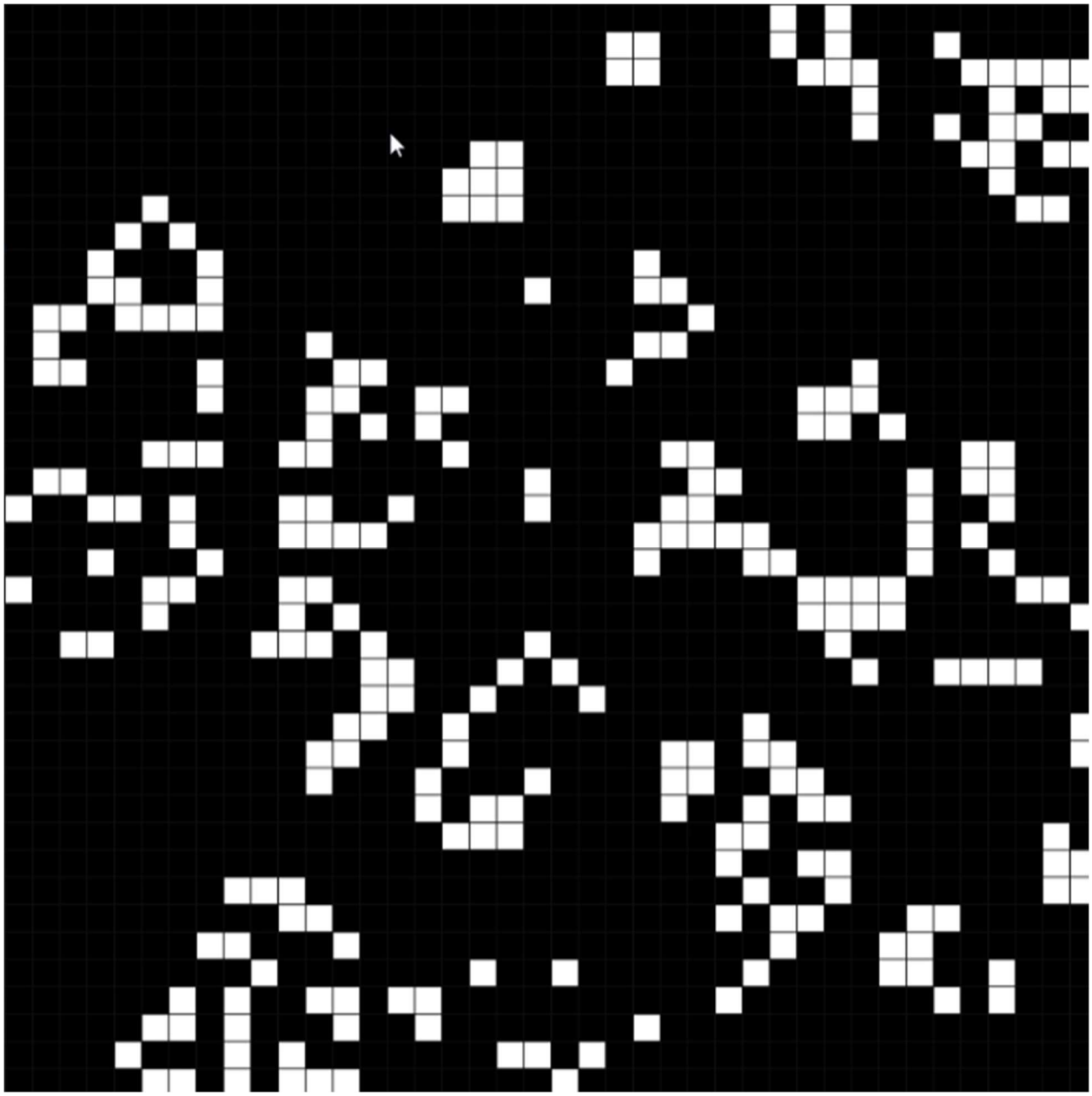


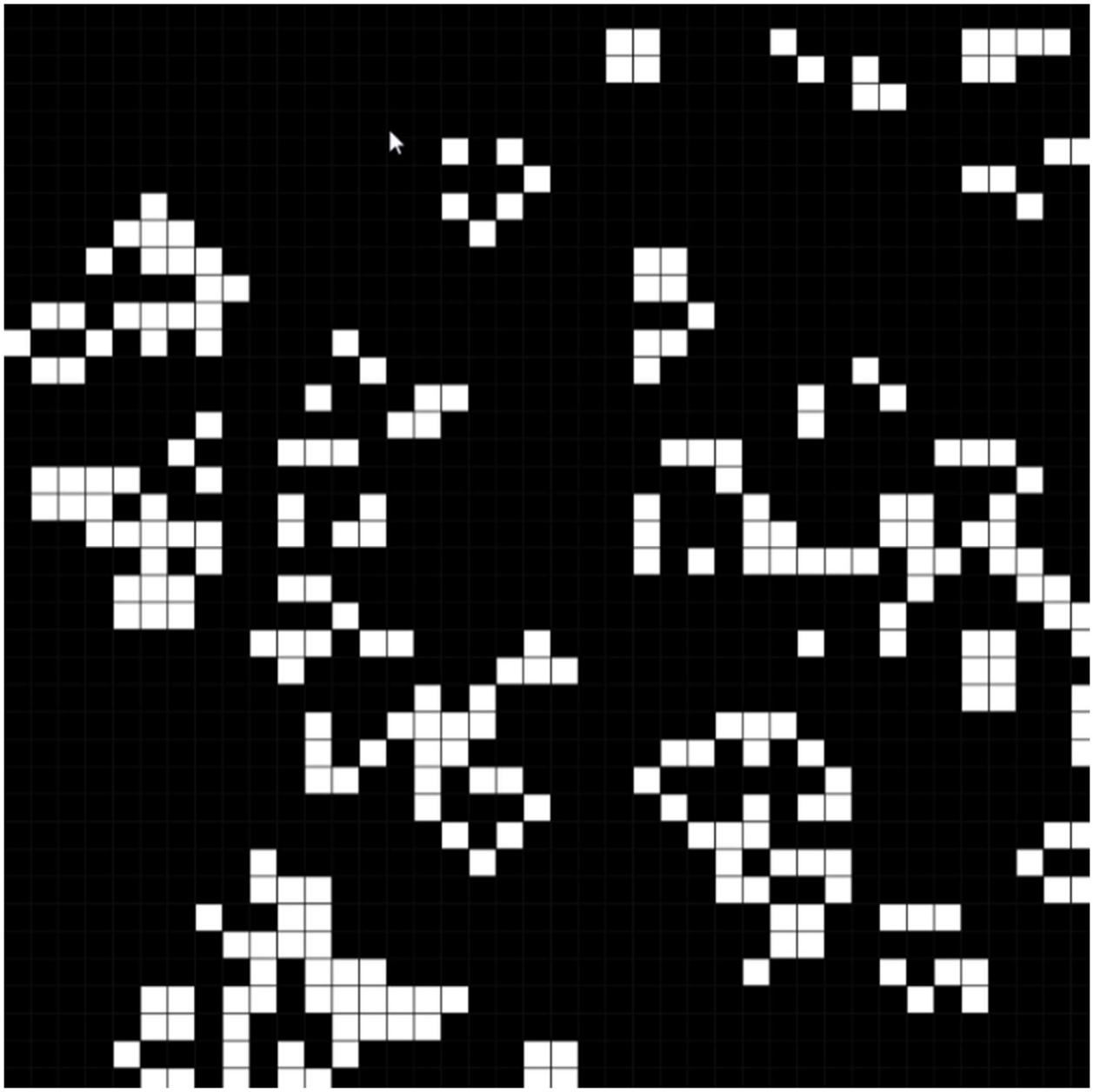












Code

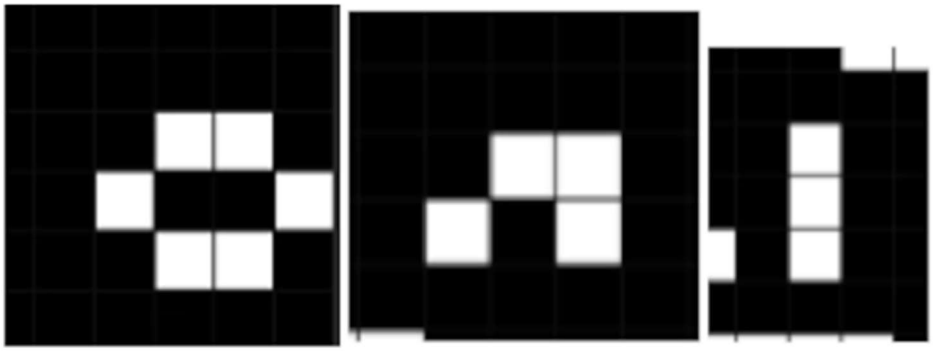
CagubcubOrticio_GameOfLife.py

```python
import pygame
import random
import multiprocessing

pygame.init()

BLACK = (15, 15, 15)
GREY = (0, 0, 0)
YELLOW = (255, 255, 255)

WIDTH, HEIGHT = 800, 800
TILE_SIZE = 20
GRID_WIDTH = WIDTH // TILE_SIZE
GRID_HEIGHT = HEIGHT // TILE_SIZE
FPS = 10

screen = pygame.display.set_mode((WIDTH, HEIGHT))

clock = pygame.time.Clock()

def gen(num):
    return set([(random.randrange(0, GRID_HEIGHT), random.randrange(0, GRID_WIDTH)) for _ in range(num)])

def draw_grid(positions):
    for position in positions:
        col, row = position
        top_left = (col * TILE_SIZE, row * TILE_SIZE)
        pygame.draw.rect(screen, YELLOW, (*top_left, TILE_SIZE, TILE_SIZE))

    for row in range(GRID_HEIGHT):
        pygame.draw.line(screen, BLACK, (0, row * TILE_SIZE), (WIDTH, row * TILE_SIZE))

    for col in range(GRID_WIDTH):
        pygame.draw.line(screen, BLACK, (col * TILE_SIZE, 0), (col * TILE_SIZE, HEIGHT))

def adjust_grid(positions):
    all_neighbors = set()
    new_positions = set()

    for position in positions:
        neighbors = get_neighbors(position)
        all_neighbors.update(neighbors)

        neighbors = list(filter(lambda x: x in positions, neighbors))

        if len(neighbors) in [2, 3]:
            new_positions.add(position)

    for position in all_neighbors:
        neighbors = get_neighbors(position)
        neighbors = list(filter(lambda x: x in positions, neighbors))

        if len(neighbors) == 3:
            new_positions.add(position)

    return new_positions

def get_neighbors(pos):
    x, y = pos
    neighbors = []
```

CagubcubOrticio_GameOfLife.py

```
43
44          neighbors = list(filter(lambda x: x in positions, neighbors))
45
46          if len(neighbors) in [2, 3]:
47              new_positions.add(position)
48
49      for position in all_neighbors:
50          neighbors = get_neighbors(position)
51          neighbors = list(filter(lambda x: x in positions, neighbors))
52
53          if len(neighbors) == 3:
54              new_positions.add(position)
55
56      return new_positions
57
58  def get_neighbors(pos):
59      x, y = pos
60      neighbors = []
61      for dx in [-1, 0, 1]:
62          if x + dx < 0 or x + dx > GRID_WIDTH:
63              continue
64          for dy in [-1, 0, 1]:
65              if y + dy < 0 or y + dy > GRID_HEIGHT:
66                  continue
67              if dx == 0 and dy == 0:
68                  continue
69
70              neighbors.append((x + dx, y + dy))
71
72      return neighbors
73
74  def play_game(positions):
75      running = True
76      while running:
77          clock.tick(FPS)
78
79          positions = adjust_grid(positions)
80
81          screen.fill(GREY)
82          draw_grid(positions)
83          pygame.display.update()
84
85          for event in pygame.event.get():
86              if event.type == pygame.QUIT:
87                  running = False
88
89  if __name__ == "__main__":
90      positions = gen(random.randrange(4, 15) * GRID_WIDTH)
91
92      game_process = multiprocessing.Process(target=play_game, args=(positions,))
93
94      game_process.start()
95
96      while True:
97          for event in pygame.event.get():
98              if event.type == pygame.QUIT:
99                  game_process.terminate()
100                 pygame.quit()
101                 exit()
```

Patterns discovered in the Simulation



still life

Space ships

oscillators