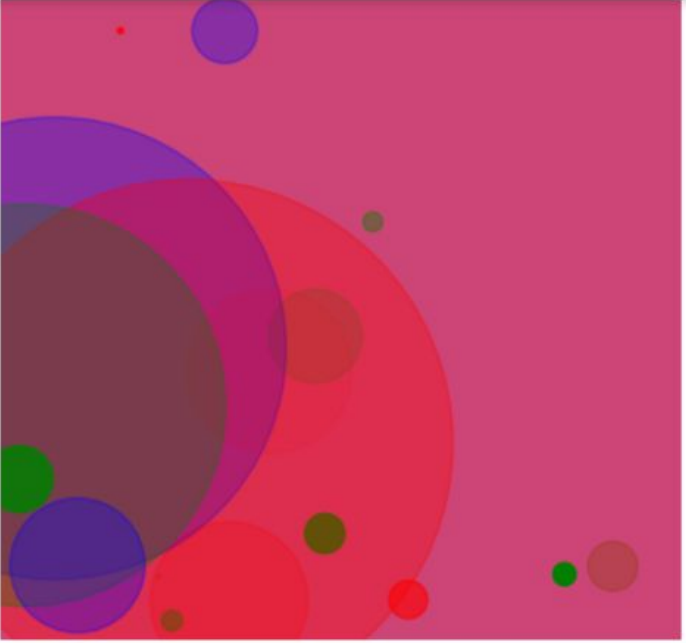
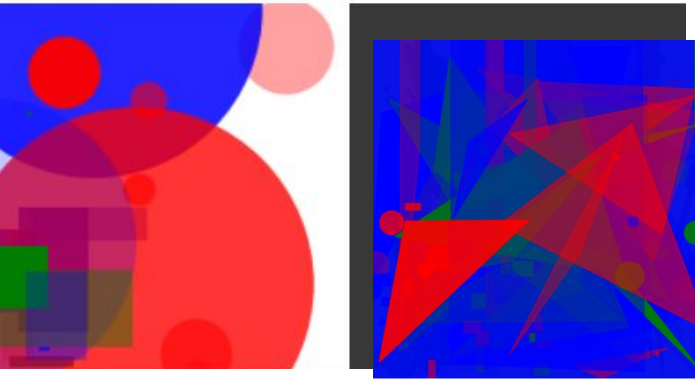




Arte evolutivo mediante algoritmos genéticos

María Carmen Aguirre Delgado

Proyecto final

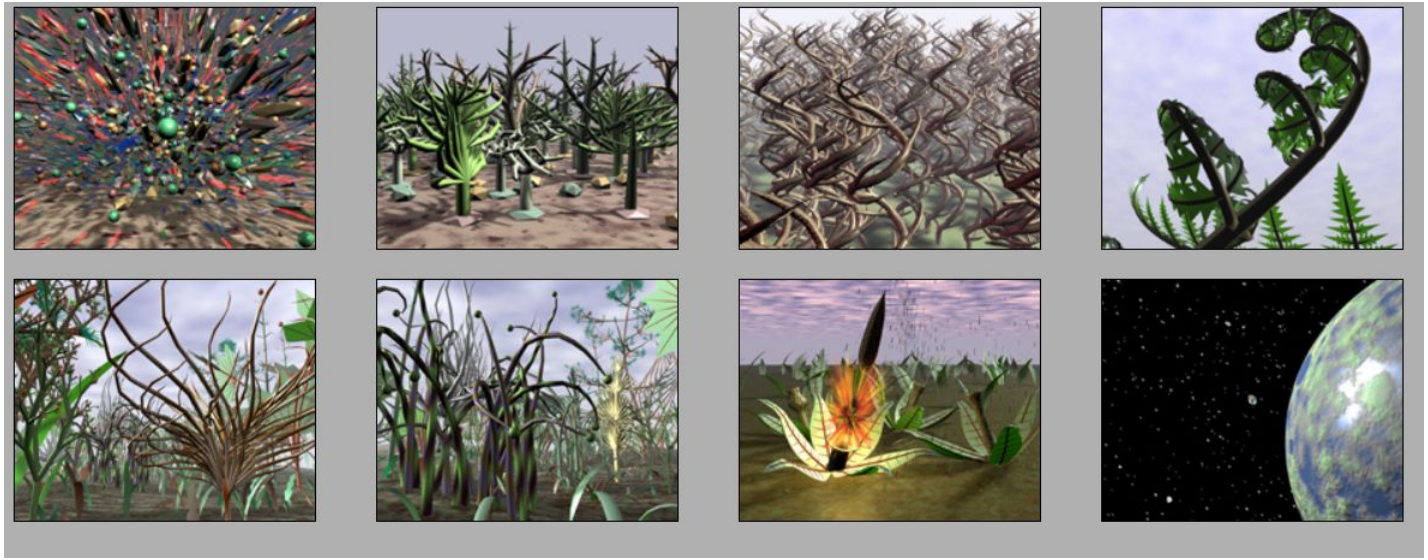
PCIC - Cómputo evolutivo

Introducción

Los algoritmos genéticos son una técnica de la Inteligencia Artificial, que simula el proceso evolutivo de los seres vivos y lo aplica a la búsqueda de soluciones y optimización, en la resolución de problemas. El arte genético es generado por computadora a partir de algoritmos genéticos.

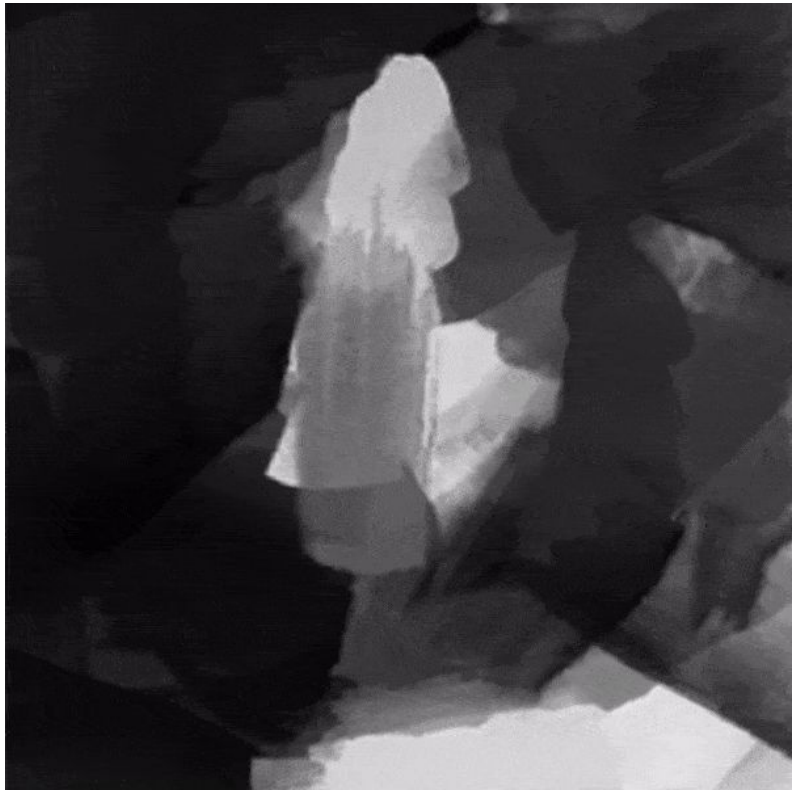
Introducción

Panspermia Karl Sims, 1990,



Se utilizó un software para crear y animar bosques de estructuras vegetales en 3D. Se utilizaron técnicas de "evolución artificial" para seleccionar interactivamente mutaciones aleatorias de formas vegetales hasta que surgió una variedad de estructuras interesantes.

Ejemplos de arte usando GA



Objetivo

Evolucionar imágenes usando algoritmos genéticos

Hay dos vertientes que podemos explorar:

- De manera no supervisada: usando medidas de simetría como funciones objetivo
- De manera supervisada: usando una imagen como objetivo

Preguntas de investigación

- ¿Es posible evolucionar imágenes **agradables a la vista humana** usando solo medidas de simetría?

- ¿Podemos **generar una imagen objetivo** a partir de figuras geométricas iniciales?

- Y si es así, ¿**qué figuras geométricas** son las mejores elecciones para lograrlo?

Implementación del AG:

- La población será una colección de imágenes
- Cada individuo es una imagen
- El cromosoma es una lista de figuras geométricas
- El fitness es la distancia (basada en píxeles) a la pintura objetivo (tenemos que minimizar esta puntuación) | Una función de simetría
- Los individuos pueden reproducirse: se crea un nueva imagen con un porcentaje de cada progenitor
- Los individuos pueden mutar, las figuras pueden moverse, cambiar de forma, cambiar de color, o cambiar de posición en el cromosoma

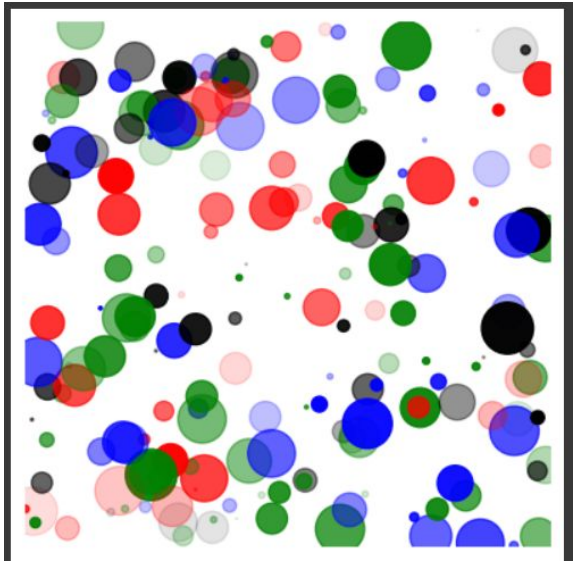
Metodología

Definición de los Individuos y Población Inicial:

- Definir la estructura de los individuos que representarán las figuras geométricas y sus características.
- Inicializar una población inicial de individuos:

  Iniciación aleatoria de colores, radios, posiciones, opacidad

  Población inicial y número de figuras por individuo fijas



Ejemplo de individuo con 200 figuras

Metodología

Mutación

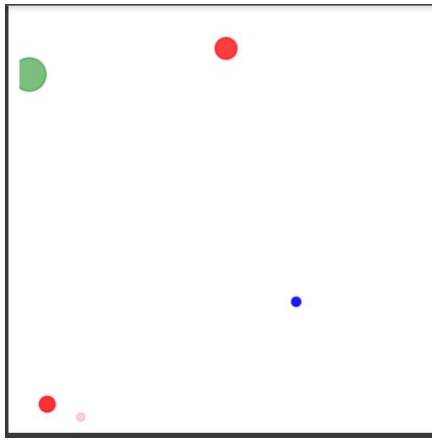
```
def mutate(individual):
    if random.random() < mutate_shape_probability:
        toolbox.mutate_shape(individual)

    if random.random() < mutate_color_probability:
        toolbox.mutate_color(individual)

    if random.random() < mutate_rotation_probability:
        toolbox.mutate_rotation(individual)

    if random.random() < remove_shape_probability:
        toolbox.remove_shape(individual)

    if random.random() < add_shape_probability:
        toolbox.add_shape(individual)

    return individual,
```

Metodología

Cruza

Selección por torneo (k=3, 10)

Cruza en un punto



```
population[0]
```

```
[('circle',
  'green',
  0.7170248150866133,
  0.5639565899864107,
  0.17460447447484817,
  0.01316852386882883),
 ('circle',
  'green',
  0.7152390382419116,
  0.06354669868779295,
  0.7698830264336329,
  0.03834652950848973),
 ('circle',
  'black',
  0.551132395369143,
  0.46097668664427693,
  0.012308560285440029,
  0.022750817072823143),
 ('circle',
  'blue',
  0.7402936846251347,
  0.5930920950532348,
  0.16438790754985555,
  0.007491708401123881),
 ('circle',
  'blue',
  0.9751474289581301,
  0.17659777290133405,
  0.3863112999594207,
  0.016523077584842227)]
```

Ejemplo de individuo: representación y visualización

Metodología

Fitness: L2

En la función `evaluate`, realiza el cálculo de la métrica L2:

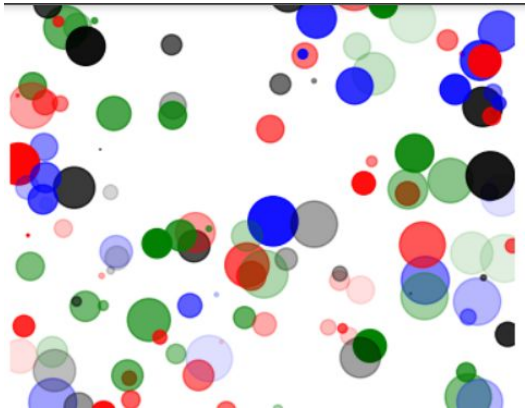
1. Las imágenes generada y objetivo se convierten a matrices de numpy.
2. Se realiza la resta elemento a elemento entre las matrices de las imágenes generada y objetivo. Esto produce una matriz de diferencias, donde cada elemento representa la diferencia entre los valores de los píxeles correspondientes en las dos imágenes.
3. Los elementos de la matriz de diferencias se elevan al cuadrado.
4. Se calcula la media de los elementos de la matriz de diferencias al cuadrado.
5. Se calcula la raíz cuadrada de la media de los elementos al cuadrado para obtener la métrica L2.

Capaz de evaluar imagenes de cualquier resolución, RBG y RGBA

Utiliza la biblioteca PILLOW para representar cada imagen

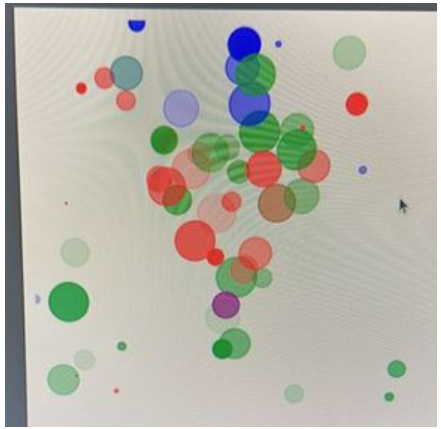
Resultados

Target

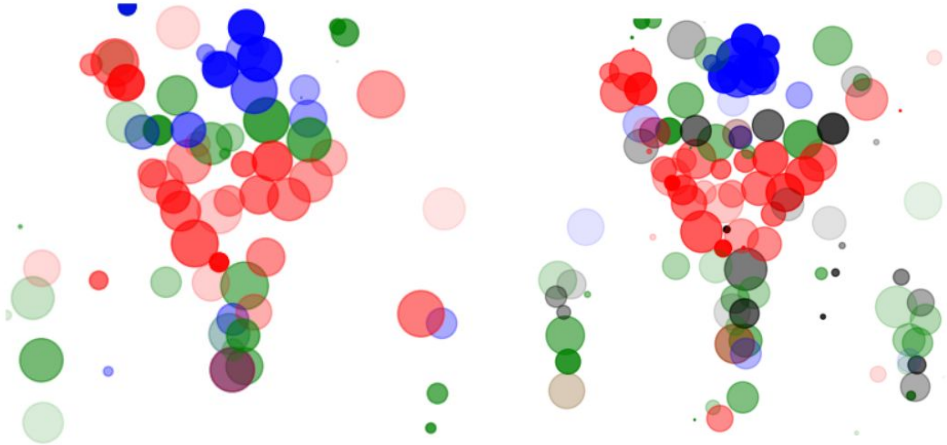


Individuos iniciales

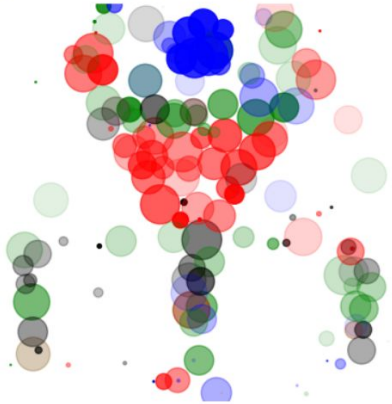
```
# Configuración del algoritmo genético
population_size = 20
num_generations =275
crossover_probability = 0.7
mutation_probability = 0.6
mutate_shape_probability = 0.0  # Probabilidad de mutación de forma
mutate_color_probability = 0.3  # Probabilidad de mutación de color
mutate_rotation_probability = 0.0  # Probabilidad de mutación de color
remove_shape_probability = 0.5# Probabilidad de mutación de remover figura
add_shape_probability = 0.3# Probabilidad de mutación de agregar figura
```



275 generaciones



+400 generaciones

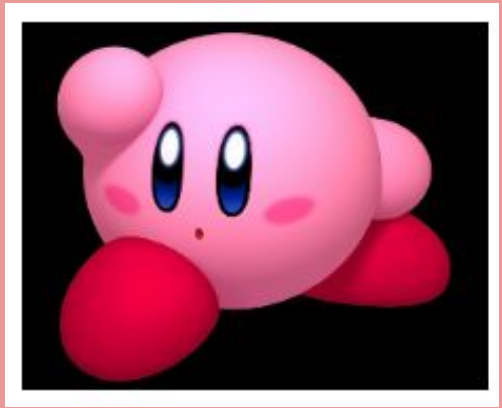


+360 generaciones

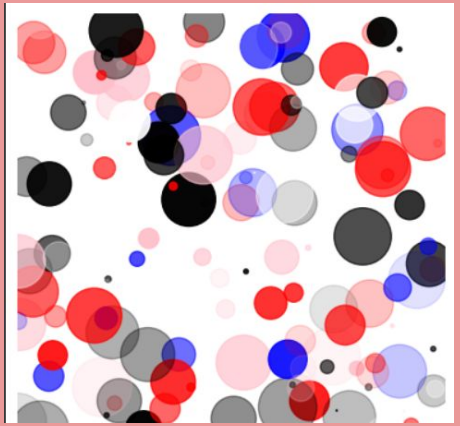
+600 generaciones

Loss inicial: 211

Loss final: 38



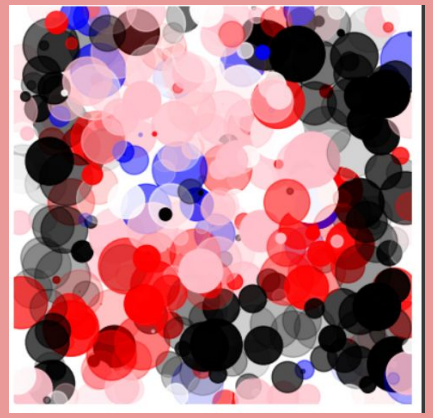
Target



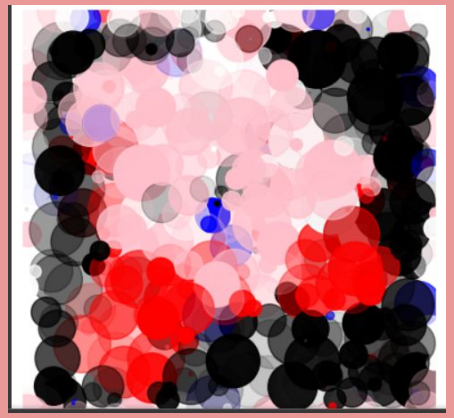
Individuos iniciales

```python
# Función para inicializar un individuo
def initialize_individual():
    return creator.Individual(generate_shape() for _ in range(150))  # Ejemplo con 150 figuras geométricas
```

```python
# Configuración del algoritmo genético
population_size = 40
num_generations =500
crossover_probability = 0.8
mutation_probability = 0.6
mutate_shape_probability = 0.0  # Probabilidad de mutación de forma
mutate_color_probability = 0.3  # Probabilidad de mutación de color
mutate_rotation_probability = 0.0  # Probabilidad de mutación de color
remove_shape_probability = 0.1# Probabilidad de mutación de remover figura
add_shape_probability = 0.7# Probabilidad de mutación de agregar figura
```



500

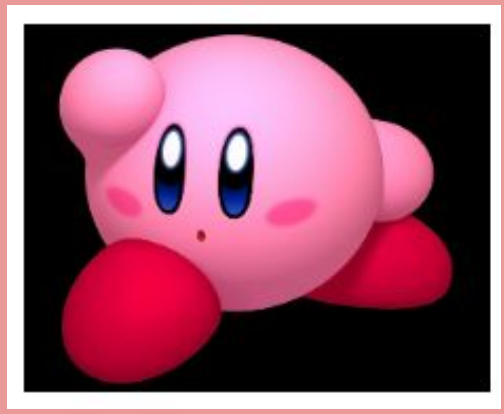


+600

+1000

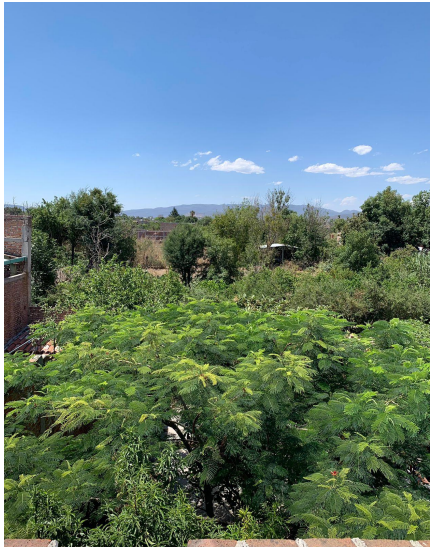
+2000

+6000

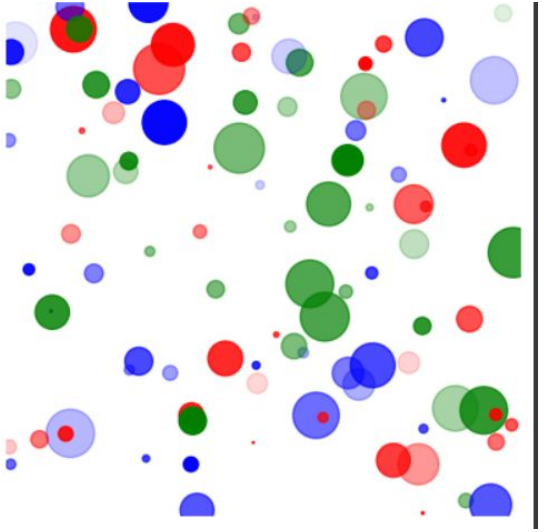


Loss ini: 160

Loss fin: 38

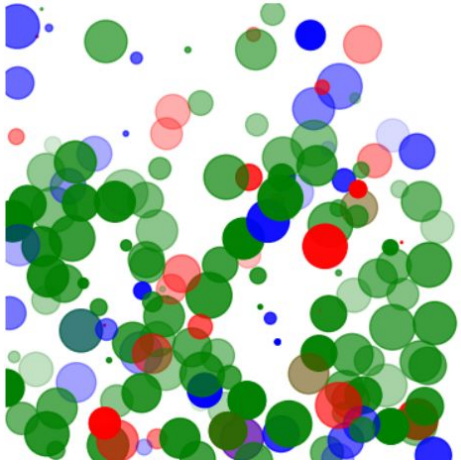


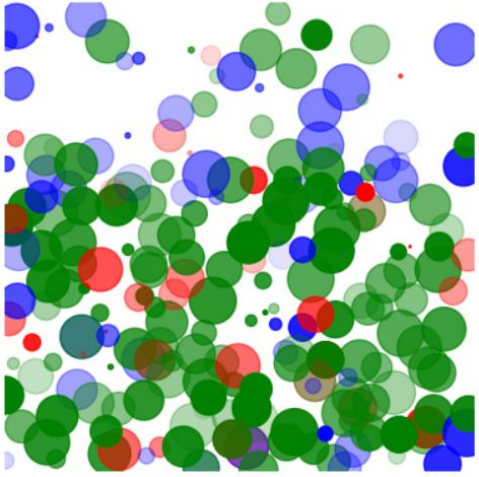
Target

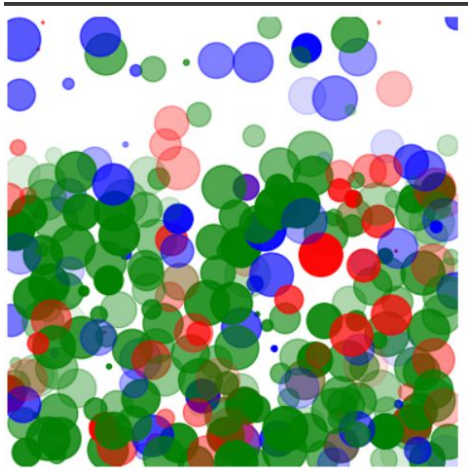


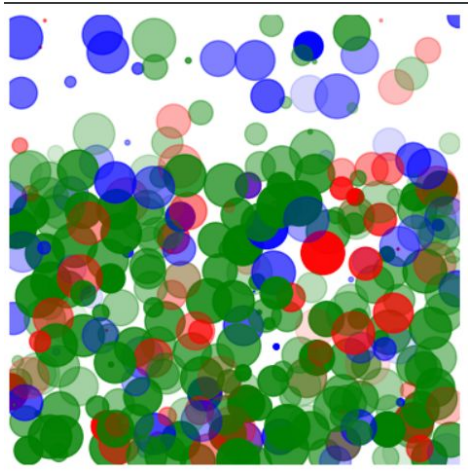
Individuos iniciales

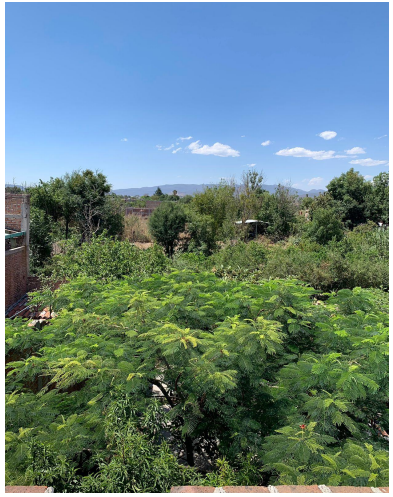
```
# Configuración del algoritmo genético
population_size = 20
num_generations =66
crossover_probability = 0.7
mutation_probability = 0.6
mutate_shape_probability = 0.0  # Probabilidad de mutación de forma
mutate_color_probability = 0.3  # Probabilidad de mutación de color
mutate_rotation_probability = 0.0  # Probabilidad de mutación de color
remove_shape_probability = 0.3# Probabilidad de mutación de remover figura
add_shape_probability = 0.6# Probabilidad de mutación de agregar figura
```











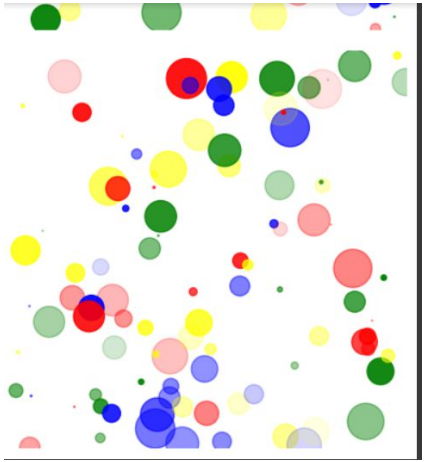
Loss ini: 126

Loss fin: 107

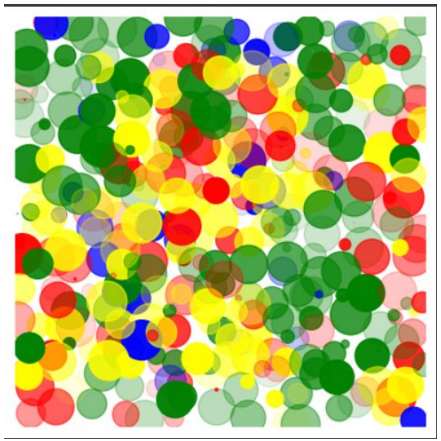
```
# Configuración del algoritmo genético
population_size = 40
num_generations =500
crossover_probability = 0.8
mutation_probability = 0.6
mutate_shape_probability = 0.0  # Probabilidad de mutación de forma
mutate_color_probability = 0.3  # Probabilidad de mutación de color
mutate_rotation_probability = 0.0  # Probabilidad de mutación de color
remove_shape_probability = 0.1# Probabilidad de mutación de remover figura
add_shape_probability = 0.7# Probabilidad de mutación de agregar figura
```

Loss inicial: 164
Loss final: 82

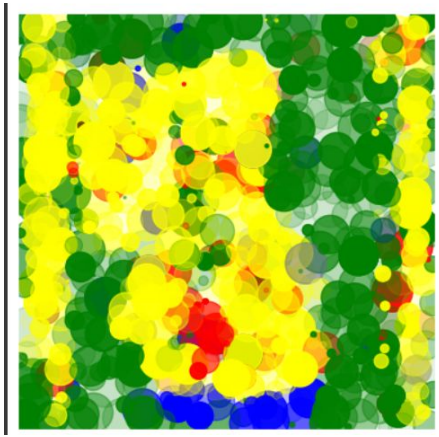
Target



Individuos iniciales



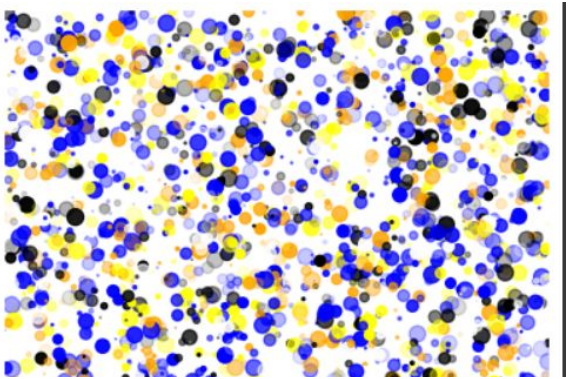
500 gen



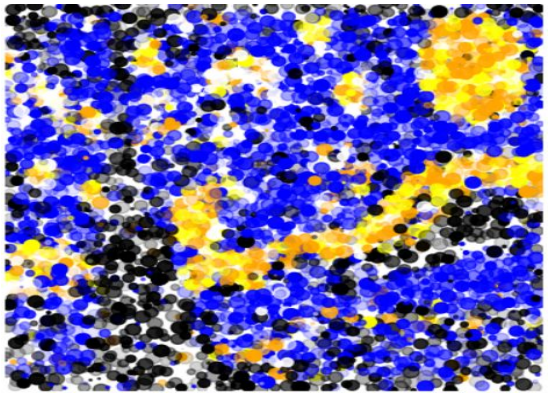
Individuo final+4000

```python
# Configuración del algoritmo genético
population_size = 100
num_generations =6000
crossover_probability = 0.8
mutation_probability = 0.6
mutate_shape_probability = 0.0  # Probabilidad de mutación de forma
mutate_color_probability = 0.3  # Probabilidad de mutación de color
mutate_rotation_probability = 0.0  # Probabilidad de mutación de color
remove_shape_probability = 0.35# Probabilidad de mutación de remover figura
add_shape_probability = 0.6# Probabilidad de mutación de agregar figura
    def initialize_individual():
        return creator.Individual(generate_shape() for _ in range(3000))
```

Target

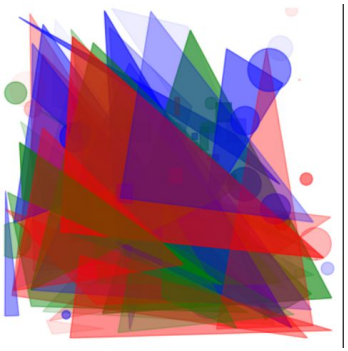


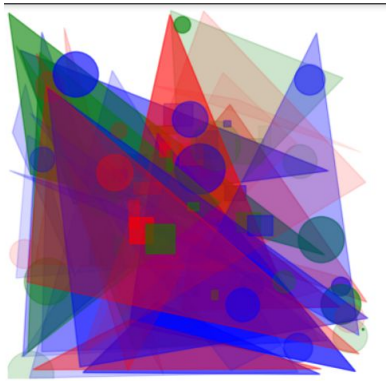
Individuos iniciales
300 pts



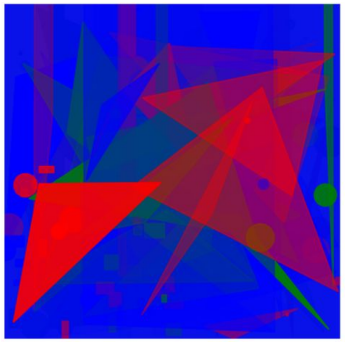
Individuo final 6000
Loss ini: 120
Loss fin: 97

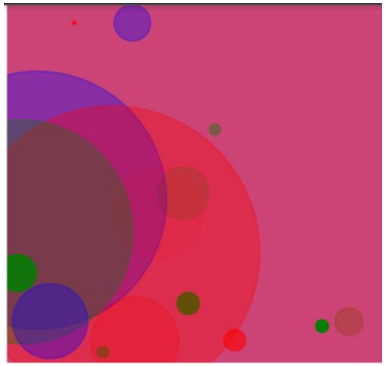
Resultados: Utilizando simetría como fitness





Ejemplo de
individuos iniciales





Individuos después
de *n* generaciones

Conclusiones

- Fue posible generar imágenes utilizando círculos aleatorios en la población de imagenes
- Se obtienen buenos resultados con suficientes generaciones
- La calidad de las imágenes depende del tamaño de las figuras iniciales
- El uso de simetría converge a una imagen monocromática en nuestra implementación

Perspectivas

- Usar más figuras por sí solas, usar combinaciones de figuras
- Explorar resultados con diferentes tamaño de figuras
- Explorar técnicas de convolución para comparación de imágenes
- En el problema de simetría: explorar más medidas de simetría, e.g. por cuadrantes
- Utilizar más de un objetivo: e.g. Colores en una imagen, estilo en otra
- Emplear probabilidades dinámicas dado algún criterio de cambio
- Añadir probabilidad de selección de colores

Gracias por su atención