gen

May 16, 2023

0.1 Enunciado

Las aplicaciones que tenés en el celular son una ventana a tu personalidad. ¿Qué dicen sobre vos? Anotá 25 aplicaciones que tenés instaladas, clasificalas y dibujalas según las instrucciones que se indican más abajo.

[ ]:
```
apps = {
        'Whatsapp': {
                'size': 10,
                'use_frecuency': 10,
                'likeability': 8,
                'type': 'Comunication'
        },
        'Discord': {
                'size': 3,
                'use_frecuency': 4,
                'likeability': 4,
                'type': 'Comunication'
        },
        'Instagram': {
                'size': 10,
                'use_frecuency': 10,
                'likeability': 8,
                'type': 'Social'
        },
        'Twitter': {
                'size': 3,
                'use_frecuency': 10,
                'likeability': 9,
                'type': 'Social'
        },
        'Pinterest': {
                'size': 3,
                'use_frecuency': 7,
                'likeability': 9,
                'type': 'Social'
        },
        'Disney+': {
                'size': 6,
                'use_frecuency': 7,
                'likeability': 9,
                'type': 'Entertainment'
        },
        'Netflix': {
                'size': 8,
                'use_frecuency': 8,
                'likeability': 7,
                'type': 'Entertainment'
        },
        'Youtube': {
                'size': 10,
                'use_frecuency': 10,
                'likeability': 10,
                'type': 'Entertainment'
        },
        'Spotify': {
                'size': 10,
                'use_frecuency': 10,
                'likeability': 10,
                'type': 'Music'
        },
        'YouTube Music': {
                'size': 3,
                'use_frecuency': 7,
                'likeability': 9,
                'type': 'Music'
        },
        'GarageBand': {
                'size': 3,
                'use_frecuency': 3,
                'likeability': 6,
                'type': 'Music'
        },
        'Adidas': {
                'size': 3,
                'use_frecuency': 3,
                'likeability': 7,
                'type': 'Sports and Health'
        },
        'SportClub': {
                'size': 3,
                'use_frecuency': 9,
                'likeability': 7,
                'type': 'Sports and Health'
        },
        'Fitness': {
                'size': 2,
                'use_frecuency': 9,
                'likeability': 10,
                'type': 'Sports and Health'
        },
        'Salud': {
                'size': 2,
                'use_frecuency': 9,
                'likeability': 10,
                'type': 'Sports and Health'
        },
        'MercadoPago': {
                'size': 4,
                'use_frecuency': 8,
                'likeability': 8,
                'type': 'Identity'
        },
        'Mi Argentina': {
                'size': 3,
                'use_frecuency': 2,
                'likeability': 0,
                'type': 'Identity'
        },
        'ACAMovil': {
                'size': 3,
                'use_frecuency': 2,
                'likeability': 0,
                'type': 'Identity'
        },
        'Wallet': {
                'size': 2,
                'use_frecuency': 8,
                'likeability': 8,
                'type': 'Identity'
        },
        'Arc': {
                'size': 3,
                'use_frecuency': 10,
                'likeability': 10,
                'type': 'Productivity'
        },
        'Drive': {
                'size': 10,
                'use_frecuency': 8,
                'likeability': 8,
                'type': 'Productivity'
        },
        'Figma': {
                'size': 3,
                'use_frecuency': 10,
                'likeability': 10,
                'type': 'Productivity'
        },
        'Github': {
                'size': 3,
                'use_frecuency': 10,
                'likeability': 10,
                'type': 'Productivity'
        },
        'Visual Studio Code': {
                'size': 3,
                'use_frecuency': 10,
                'likeability': 10,
                'type': 'Productivity'
        },
        'Campus Di Tella': {
                'size': 3,
                'use_frecuency': 6,
                'likeability': 4,
                'type': 'Productivity'
        }
}
```

[ ]: 25

[ ]:
```python
from pprint import pprint
from IPython.display import display, Markdown, Image

def print_md(string):
    display(Markdown(string))

app_list = '\n - '.join(list(apps.keys()))

print_md(f'## Apps: \n - {app_list}')
```

0.2 Apps:

- Whatsapp
- Discord
- Instagram
- Twitter
- Pinterest
- Disney+
- Netflix

- Youtube
- Spotify
- YouTube Music
- GarageBand
- Adidas
- SportClub
- Fitness
- Salud
- MercadoPago
- Mi Argentina
- ACAMovil
- Wallet
- Arc
- Drive
- Figma
- Github
- Visual Studio Code
- Campus Di Tella

0.3 Primera parte: Clasificación

```
[ ]: # Print app types

     types = set([app['type'] for app in apps.values()])

     print_md(f'### App types')

     group_by_type = {}

     for app_name, app in apps.items():
         group_by_type.setdefault(app['type'], []).append(app_name)

     for app_type, app_list in group_by_type.items():
         print_md(f'#### {app_type}')
         subapp_list = "\n - ".join(app_list)
         print_md(f' - {subapp_list}')
```

0.4 App types

0.4.1 Comunication

- Whatsapp
- Discord

0.4.2 Social

- Instagram
- Twitter
- Pinterest

0.4.3 Entertainment

- Disney+
- Netflix
- Youtube

0.4.4 Music

- Spotify
- YouTube Music
- GarageBand

0.4.5 Sports and Health

- Adidas
- SportClub
- Fitness
- Salud

0.4.6 Identity

- MercadoPago
- Mi Argentina
- ACAMovil
- Wallet

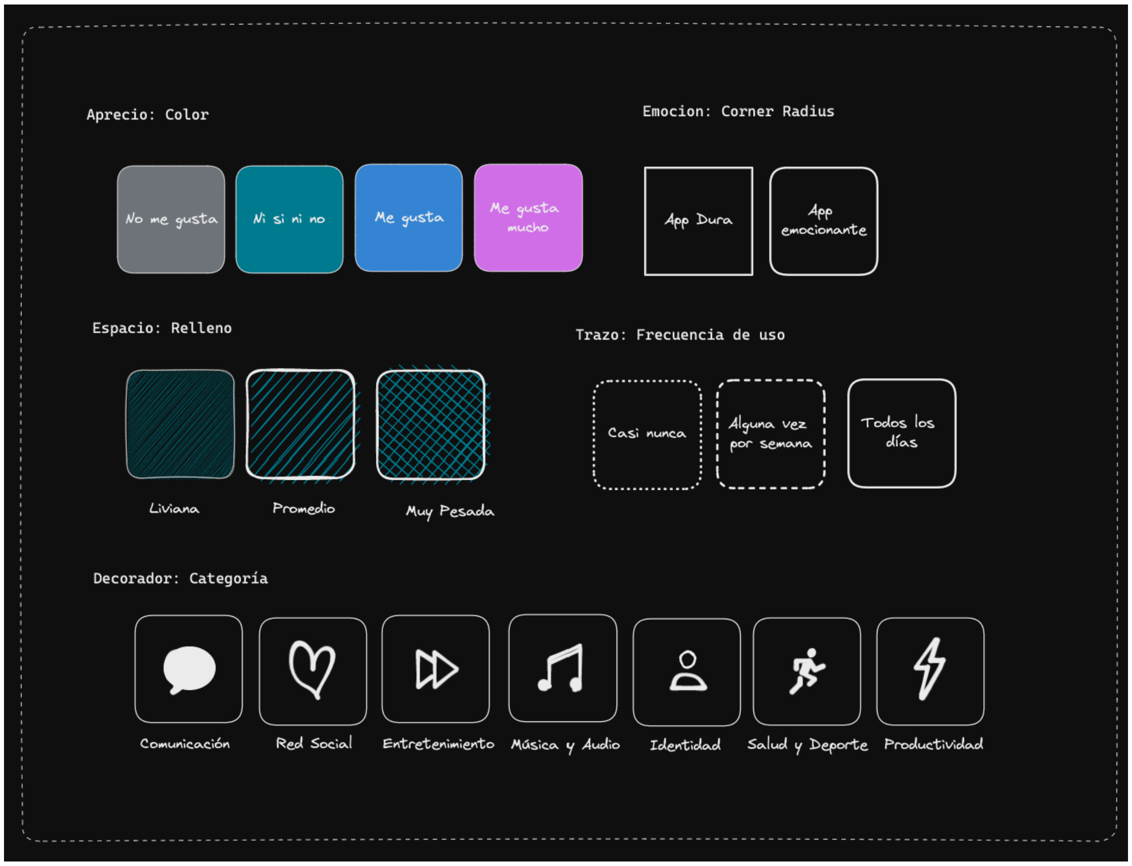
0.4.7 Productivity

- Arc
- Drive
- Figma
- Github
- Visual Studio Code
- Campus Di Tella

0.5 Desarrollo de un pimer sistema de diseño a mano

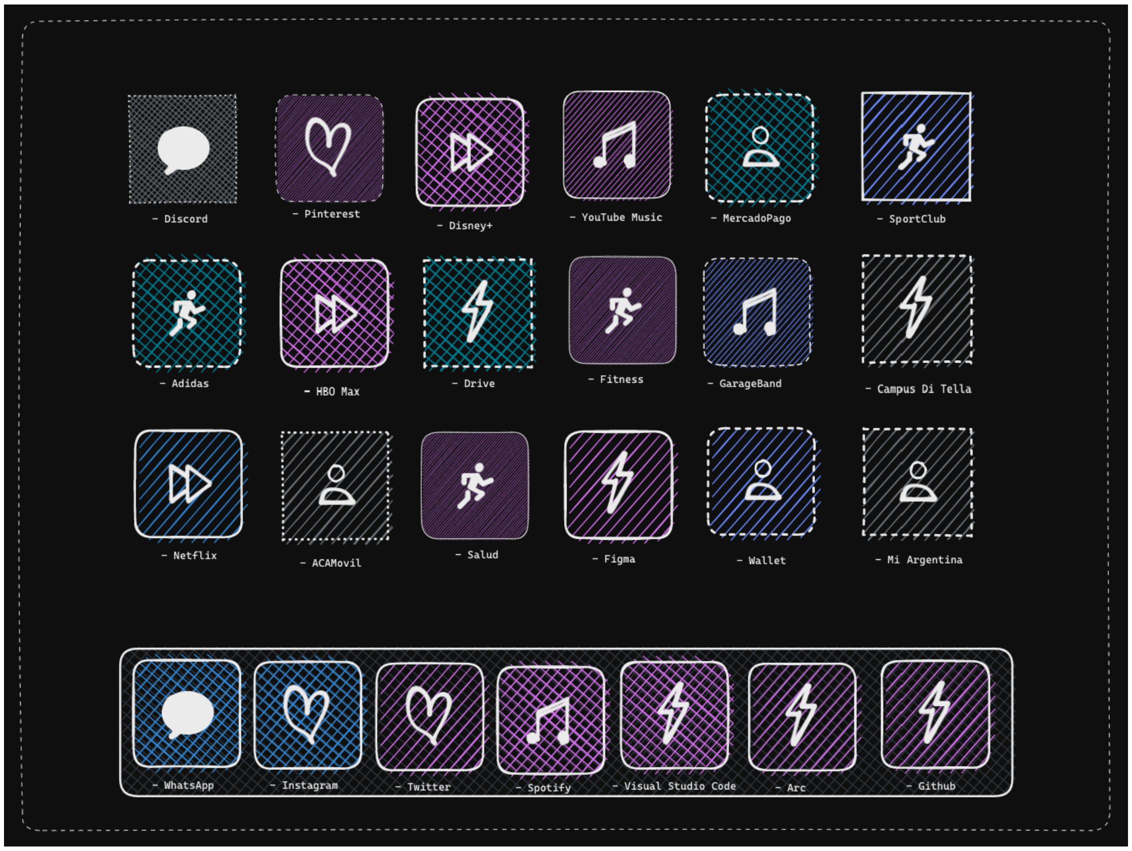
0.5.1 Referencia

```
[ ]: display(Image(filename='draw_ref.png'))
```



0.5.2 Aplicado a mis apps

```
[ ]: display(Image(filename='draw.png'))
```



0.6 Segunda parte: Un sistema de diseño parametrizado general

0.6.1 Idea

La idea de tener un sistema de diseño parametrizado es que podamos generar distintas imágenes a partir de distintos valores de c.

Para ello una primera idea fue el Mandelbroth Set, un fractal que se genera a partir de la siguiente fórmula:

$$z_{n+1} = z_n^2 + c$$

donde $z_0 = 0$ y c es un número complejo.

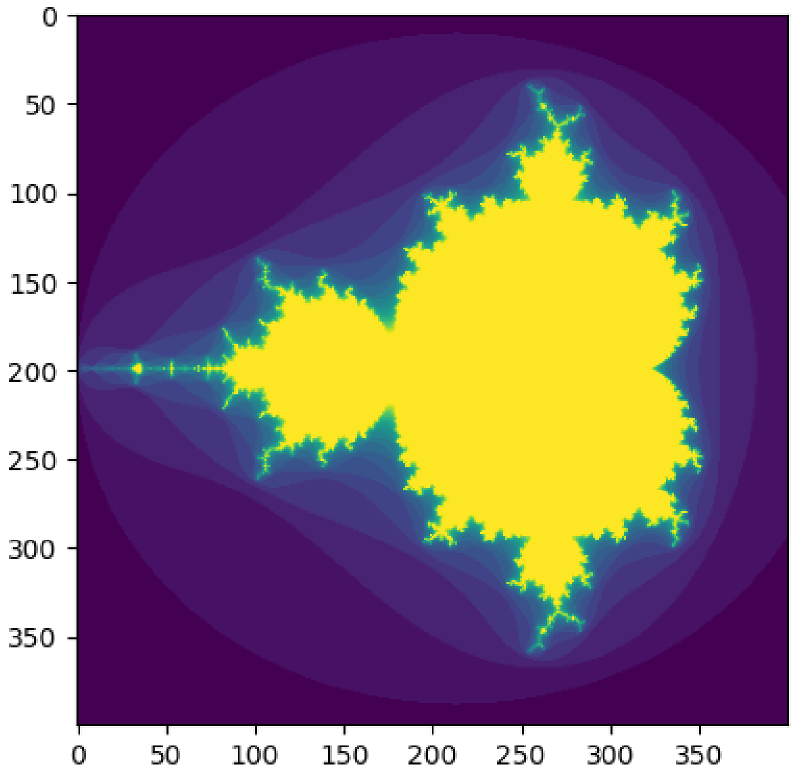
Entonces podemos generar distintas imágenes a partir de distintos valores de c, y combinar otros factores, como el color de la imagen o la resolución del fractal para aprovechar en nuestro sistema de diseño.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
```

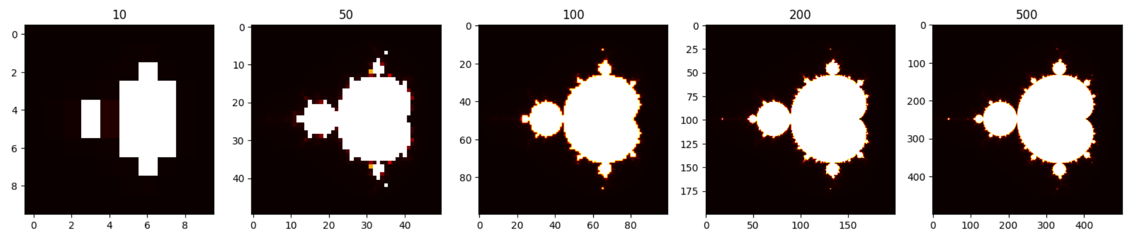
Algunos ejemplos de imágenes generadas con distintos valores de c:

```
[ ]: # A simple mandelbrot set generator

     def mandelbrot(h, w, maxit=20) -> np.ndarray:
         """Returns an image of the Mandelbrot fractal of size (h,w)."""
         y,x = np.ogrid[ -1.4:1.4:h*1j, -2:0.8:w*1j ]
         c = x+y*1j
         z = c
         divtime = maxit + np.zeros(z.shape, dtype=int)

         for i in range(maxit):
             z = z**2 + c
             diverge = z*np.conj(z) > 2**2            # who is diverging
             div_now = diverge & (divtime==maxit)  # who is diverging now
             divtime[div_now] = i                  # note when
             z[diverge] = 2                        # avoid diverging too much

         return divtime

     plt.imshow(mandelbrot(400,400))
```

```
[ ]: <matplotlib.image.AxesImage at 0x7f83a8ddf520>
```



[ ]:
```
fig, ax = plt.subplots(1, 5, figsize=(20, 20))

for i, size in enumerate([10, 50, 100, 200, 500]):
    ax[i].imshow(mandelbrot(size,size,500), cmap='hot')
    ax[i].set_title(f'{size}')

plt.show()
```



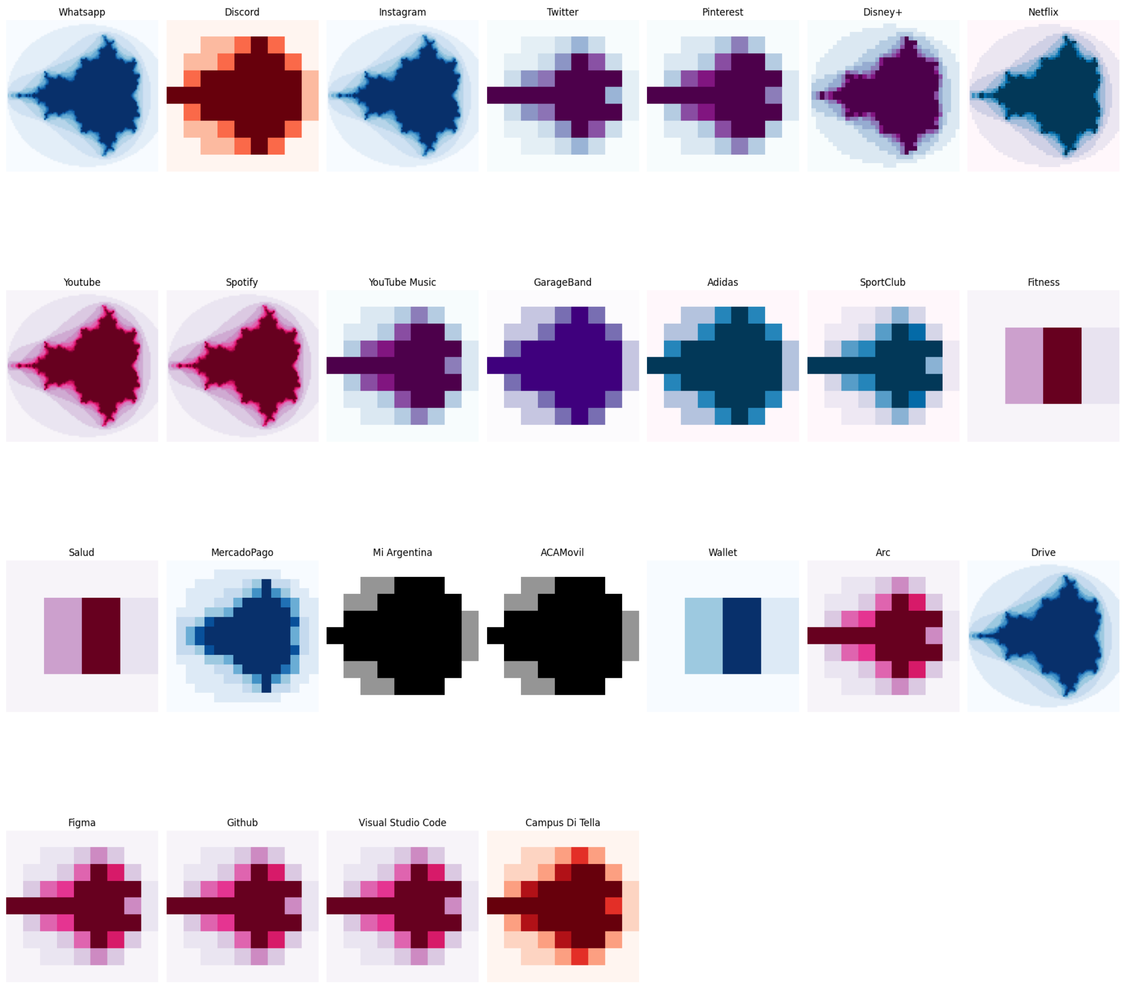
A partir de esto planteamos un sistema de diseño parametrizado que nos permita generar distintas imágenes a partir de distintos valores de c.

[ ]:
```
# Generate design system based on:

"""
- App Size: Small, Medium, Large (0, 10)
- App Type: Social, Productivity, Entertainment, etc.
- App Frequency of use: Daily, Weekly, Monthly (0, 10)
- App Likeability: (0, 10)

"""

def gen_mandelbrot_item(
        size: int,
        use_frecuency: int,
        likeability: int,
        type: str
) -> np.ndarray:
        """_summary_

        Args:
            size (int): Maps to mandelbrot size (h,w)
            use_frecuency (int): maps to mandelbrot maxit
            likeability (int): maps to color map
            type (str): maps to rotation of image
        """

        return mandelbrot(size * size, size * size, use_frecuency)
```

```
[ ]: # Add cmap value based on likeability

cmaps_dict = {
        0: 'Greys',
        1: 'Greens',
        2: 'YlGn',
        3: 'OrRd',
        4: 'Reds',
        5: 'Oranges',
        6: 'Purples',
        7: 'PuBu',
        8: 'Blues',
        9: 'BuPu',
        10: 'PuRd'
}

for app in apps:
        apps[app]['cmap'] = cmaps_dict[apps[app]['likeability']]
```

```
[ ]: # Generate every app image

for app in apps:
        apps[app]['mandelbrot'] = gen_mandelbrot_item(
                apps[app]['size'],
                apps[app]['use_frecuency'],
                apps[app]['likeability'],
                apps[app]['type']
        )

# Plot every app image

# 3 row of 6 images
# plus 1 row of 7 images

# 25 images in total

fig, axs = plt.subplots(4, 7, figsize=(20, 20))

for i, app in enumerate(apps):
        axs[i//7, i%7].imshow(apps[app]['mandelbrot']
        , cmap=apps[app]['cmap'])
        axs[i//7, i%7].axis('off')
        axs[i//7, i%7].set_title(app)

axs[3, 4].axis('off')
axs[3, 4].set_title('')
```

```
axs[3, 5].axis('off')
axs[3, 5].set_title('')
axs[3, 6].axis('off')
axs[3, 6].set_title('')
fig.tight_layout()

fig.savefig('mandelbrot.svg', dpi=300, bbox_inches='tight', pad_inches=0)


plt.show()
```



[ ]:
```
!mkdir -p mandelbrot_images
```

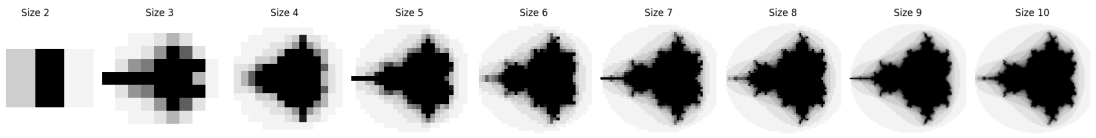
[ ]:
```
!rm -rf mandelbrot_images/*
```

```
[ ]: # Generate for each app an SVG file with the mandelbrot image

     for app in apps:
             # apps[app]['mandelbrot'] is a Numpy array

             # Use matplotlib to generate the SVG

             fig, ax = plt.subplots(figsize=(1, 1))

             ax.imshow(apps[app]['mandelbrot']
             , cmap=apps[app]['cmap'])
             ax.axis('off')


             # Save the SVG file

             fig.savefig(f'mandelbrot_images/{app}.svg', format='svg', dpi=300,␣
      ↪bbox_inches='tight', pad_inches=0)

             # Close the figure

             plt.close(fig)
```

Sistema de Referencia para esta primera idea

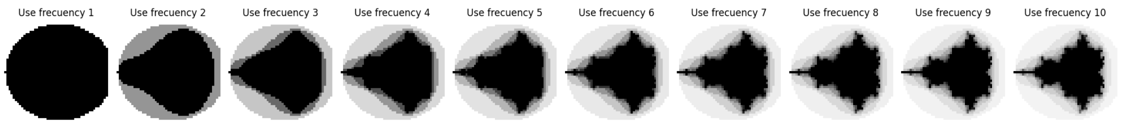
El tamaño de la app afecta el tamaño del fractal

```
[ ]: # Reference system

     # Plot an example of a mandelbrot image for reference

     # Size affects the number of iterations

     fig, ax = plt.subplots(1, 9, figsize=(20, 20))
     for s in range(2, 11):
             ax[s-2].imshow(gen_mandelbrot_item(s, 10, 10, 'Reference'),␣
      ↪cmap='Greys')
             ax[s-2].axis('off')
             ax[s-2].set_title(f'Size {s}')

     fig.tight_layout()

     plt.show()
```



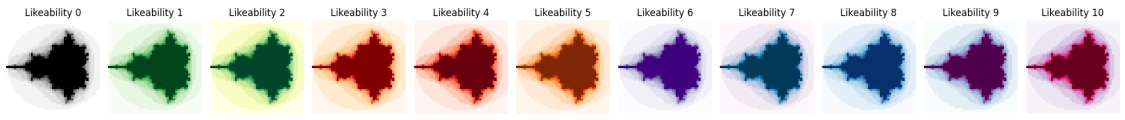
La frecuencia de uso de la app afecta la cantidad de iteraciones del fractal

```
[ ]: # Use frecuency affects the number of iterations

fig, ax = plt.subplots(1, 10, figsize=(20, 20))

for f in range(1, 11):
        ax[f-1].imshow(gen_mandelbrot_item(7, f, 10, 'Reference'), cmap='Greys')
        ax[f-1].axis('off')
        ax[f-1].set_title(f'Use frecuency {f}')

fig.tight_layout()

plt.show()
```



El “aprecio” por la app afecta el color del fractal

```
[ ]: # Likeability affects the color map

fig, ax = plt.subplots(1, 11, figsize=(20, 20))

for l in range(0, 11):
        ax[l].imshow(gen_mandelbrot_item(7, 10, l, 'Reference'),␣
  ↪cmap=cmaps_dict[l])
        ax[l].axis('off')
        ax[l].set_title(f'Likeability {l}')

fig.tight_layout()

plt.show()
```



[ ]:
```
types = {a['type'] for a in apps.values()}
types
```

[ ]:
```
{'Comunication',
 'Entertainment',
 'Identity',
 'Music',
 'Productivity',
 'Social',
 'Sports and Health'}
```

0.6.2 Ampliar el sistema de diseño con distintos fractales

El Mandelbrot Set es un ejemplo particular del Julia Set, por lo que ademas podríamos parametrizar el tipo de fractal que queremos generar.

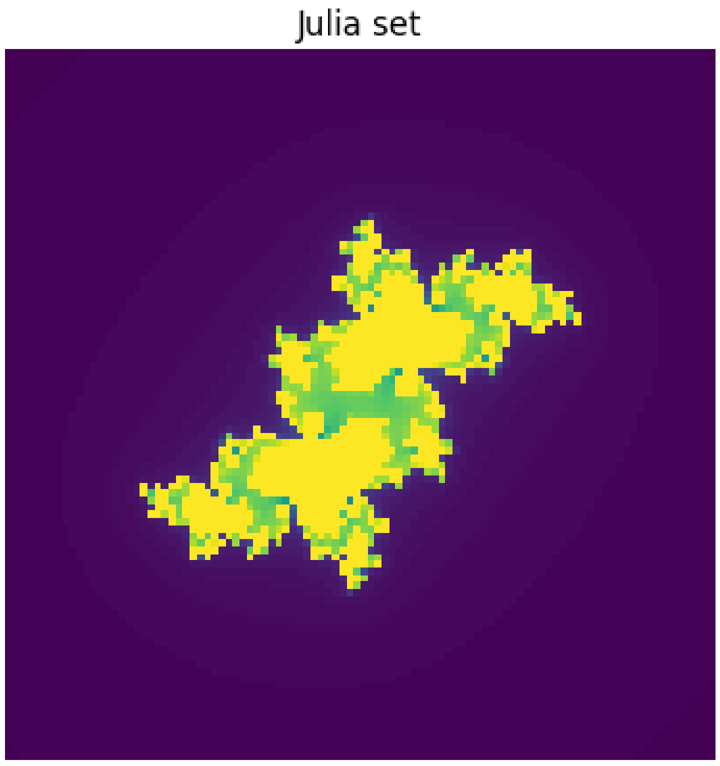
[ ]:
```
# Julia Set

def julia_set_fractal(c, n=100, thresh=50):
        """
        Copy-pasted from https://matplotlib.org/stable/gallery/
 ↪lines_bars_and_markers/fill.
 ↪html#sphx-glr-gallery-lines-bars-and-markers-fill-py
        """
        def julia(z, c):
                return z**2 + c

        m = np.zeros((n, n))
        for i, x in enumerate(np.linspace(-2, 2, n)):
                for j, y in enumerate(np.linspace(-2, 2, n)):
                        z = complex(x, y)
                        t = 0
                        while abs(z) < thresh and t < n:
                                z = julia(z, c)
                                t += 1
                        m[j, i] = t

        return m
```

[ ]:
```
# Generate design system based on:

"""
- App Size: Small, Medium, Large (0, 10)
- App Type: Social, Productivity, Entertainment, etc.
- App Frequency of use: Daily, Weekly, Monthly (0, 10)
- App Likeability: (0, 10)

"""

def gen_julia_item(
        size: int,
        use_frecuency: int,
        likeability: int,
        t: str
) -> np.ndarray:
        """_summary_

        Args:
            size (int): Maps to mandelbrot size (h,w)
            use_frecuency (int): maps to mandelbrot maxit
            likeability (int): maps to color map
            type (str): maps to the type of fractal
        """

        #return mandelbrot(size * size, size * size, use_frecuency)


        complex_types = {
                'Comunication': complex(0.285, 0.01),
                'Entertainment': complex(-0.8, 0.156),
                'Identity': complex(-0.4, 0.6),
                'Music': complex(-0.1, -0.732),
                'Productivity': complex(-0.9, 0),
                'Social': complex(-0.215, -0.65),
                'Sports and Health': complex(0.73, -0.73)
        }

        complex_n = complex_types[t]


        """ return julia_set_fractal(
                complex_n,
                n=size * size,
                thresh=use_frecuency
        ) """


        return julia_set_fractal(
                complex_n,
                n= size * size + (likeability*likeability if likeability else␣
 ↪1),
                thresh=use_frecuency
        )
```

Un ejemplo de un fractal generado por el Julia Set a partir de

$$c = -0.1 + 0.65i$$

[ ]:
```
# plot an example

fig, ax = plt.subplots()
ax.imshow(julia_set_fractal(complex(-0.1, 0.65)))
ax.axis('off')
ax.set_title("Julia set")
plt.show()
```



Aprovechando entonces el sistema de diseño que ya tenemos, podemos generar distintos fractales en función del tipo de app que queremos representar.

Para ello elegí diferentes números complejos y los asocié a distintos tipos de apps.

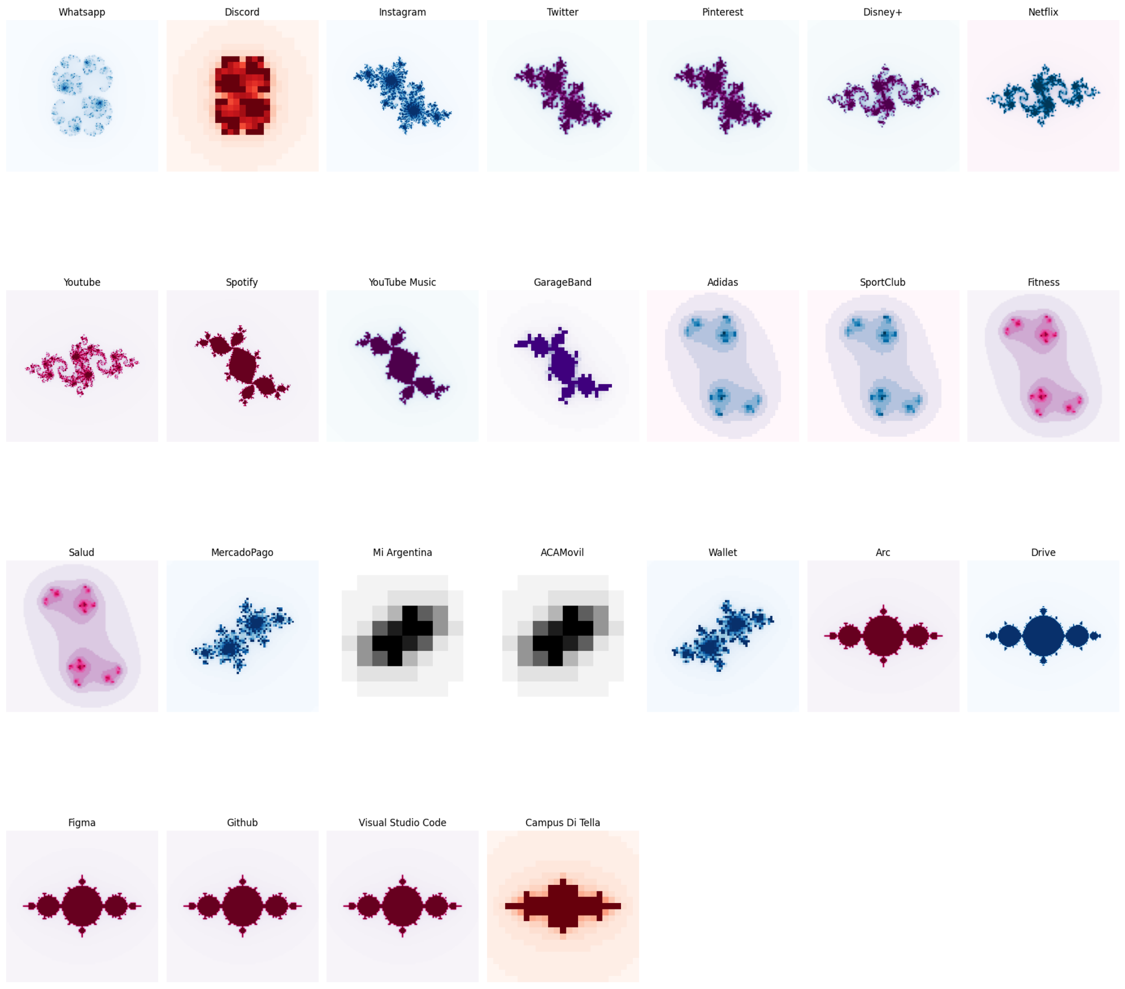
App	Número Complejo
Comunication	0.285 + 0.01i
Entertainment	-0.8 + 0.156i
Identity	-0.4 + 0.6i

App	Número Complejo
Music	-0.1 - 0.732i
Productivity	-0.9 + 0i
Social	-0.215 - 0.65i
Sports and Health	0.73 - 0.73i

De esta forma llegamos a un sistema de diseño parametrizado que nos permite generar distintos fractales en función de distintos tipos de apps.

```
[ ]: for app in apps:
             apps[app]['julia'] = gen_julia_item(
                     apps[app]['size'],
                     apps[app]['use_frecuency'],
                     apps[app]['likeability'],
                     apps[app]['type']
             )

     # Plot every app image

     # 3 row of 6 images
     # plus 1 row of 7 images

     # 25 images in total

     fig, axs = plt.subplots(4, 7, figsize=(20, 20))

     for i, app in enumerate(apps):
             axs[i//7, i%7].imshow(apps[app]['julia']
             , cmap=apps[app]['cmap'])
             axs[i//7, i%7].axis('off')
             axs[i//7, i%7].set_title(app)

             axs[i//7, i%7]



     axs[3, 4].axis('off')
     axs[3, 4].set_title('')
     axs[3, 5].axis('off')
     axs[3, 5].set_title('')
     axs[3, 6].axis('off')
     axs[3, 6].set_title('')
     fig.tight_layout()

     #fig.savefig('julia.svg', dpi=300, bbox_inches='tight', pad_inches=0)
```

```
plt.show()
```

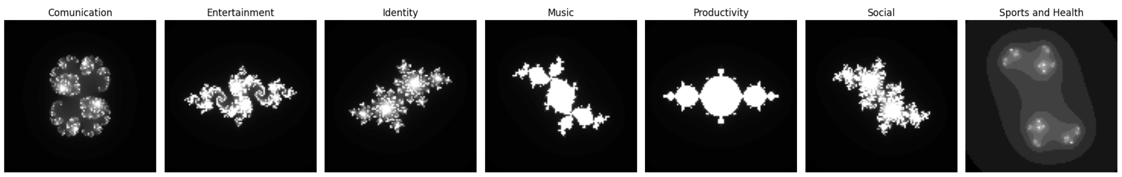


[ ]:
```
!mkdir julia_images
```

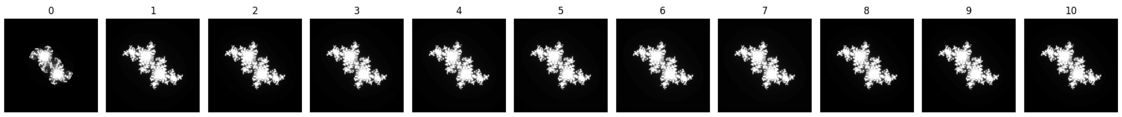
```
mkdir: julia_images: File exists
```

[ ]:
```
# Save every app image

for app in apps:

        # Create the figure

        fig, ax = plt.subplots(figsize=(20, 20))

        # Plot the image
        ax.imshow(apps[app]['julia']
        , cmap=apps[app]['cmap'])
        ax.axis('off')


        # Save the SVG file

        fig.savefig(f'julia_images/{app}.svg', format='svg', dpi=300,␣
 ↪bbox_inches='tight', pad_inches=0)

        # Close the figure

        plt.close(fig)
```

[ ]:
```
# Generate the design system

# The type of the app affects de complex number

complex_types = {
                'Comunication': complex(0.285, 0.01),
                'Entertainment': complex(-0.8, 0.156),
                'Identity': complex(-0.4, 0.6),
                'Music': complex(-0.1, -0.732),
                'Productivity': complex(-0.9, 0),
                'Social': complex(-0.215, -0.65),
                'Sports and Health': complex(0.73, -0.73)
        }

fig, ax = plt.subplots(1, 7, figsize=(20, 20))

for i, t in enumerate(complex_types):
        ax[i].imshow(julia_set_fractal(complex_types[t]), cmap='gray')
        ax[i].axis('off')
        ax[i].set_title(t)

fig.tight_layout()

fig.savefig('julia_types.svg', format='svg', dpi=300, bbox_inches='tight',␣
 ↪pad_inches=0)

plt.show()
```

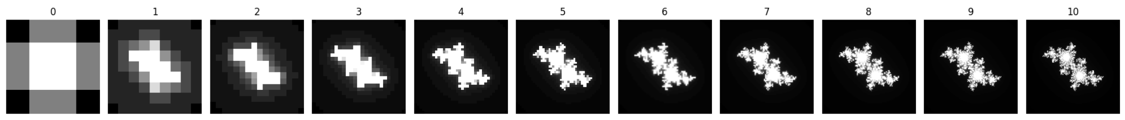


```
[ ]: # The use frecuency of the app affects the max iterations of the fractal

fig, ax = plt.subplots(1, 11, figsize=(20, 20))

for i, s in enumerate(range(0, 11)):

        ax[i].imshow(julia_set_fractal(complex_types['Social'], n=100,␣
 ↪thresh=(s+1)), cmap='gray')
        ax[i].axis('off')
        ax[i].set_title(s)

fig.tight_layout()

fig.savefig('julia_use_frecuency.svg', format='svg', dpi=300,␣
 ↪bbox_inches='tight', pad_inches=0)

plt.show()
```

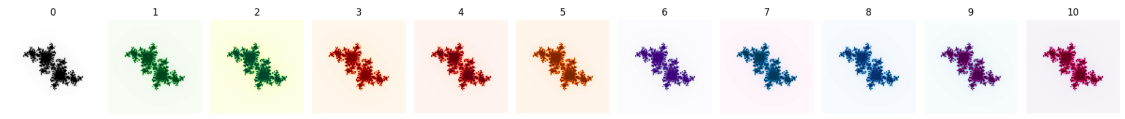


```
[ ]: # The size and the likeablility of the app affect the size of the image

fig, ax = plt.subplots(1, 11, figsize=(20, 20))

for i, s in enumerate(range(0, 11)):
        ax[i].imshow(julia_set_fractal(complex_types['Social'], n=(s+2) ** 2),␣
 ↪cmap='gray')
        ax[i].axis('off')
        ax[i].set_title(s)

fig.tight_layout()

fig.savefig('julia_size.svg', format='svg', dpi=300, bbox_inches='tight',␣
 ↪pad_inches=0)
```

```
plt.show()
```

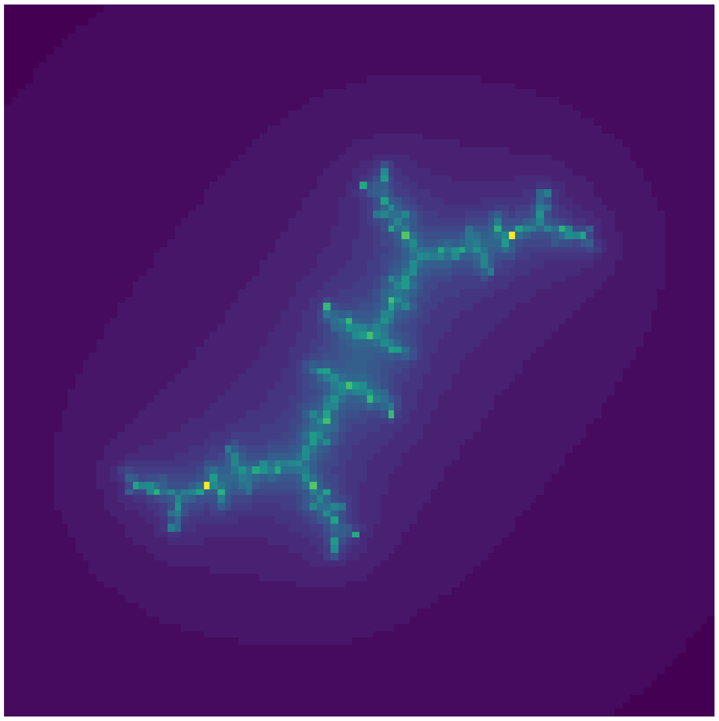


[ ]:
```
# The likeability of the app affects the color map

fig, ax = plt.subplots(1, 11, figsize=(20, 20))

for i, s in enumerate(range(0, 11)):
        ax[i].imshow(julia_set_fractal(complex_types['Social'], n=100,␣
 ↪thresh=100), cmap=cmaps_dict[s])
        ax[i].axis('off')
        ax[i].set_title(s)

fig.tight_layout()

fig.savefig('julia_likeability.svg', format='svg', dpi=300,␣
 ↪bbox_inches='tight', pad_inches=0)

plt.show()
```



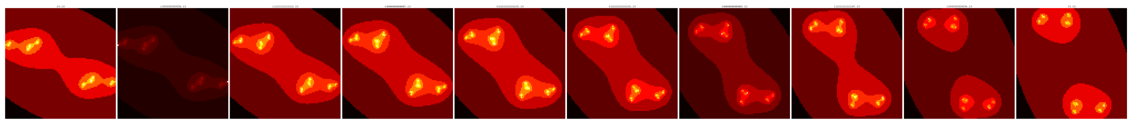
[ ]:
```
fig, ax = plt.subplots()
ax.imshow(julia_set_fractal(complex(-0.1, 1)))
ax.axis('off')
ax.set_title("Julia set")
plt.show()
```

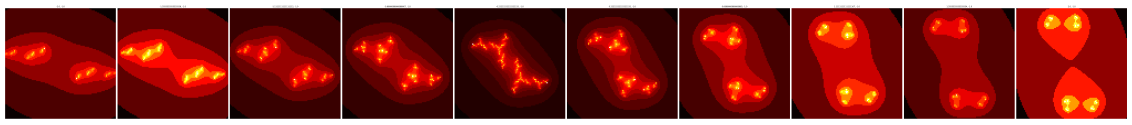
Julia set

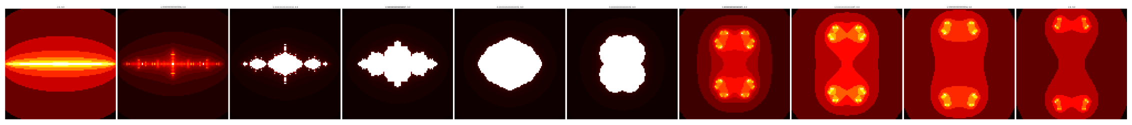


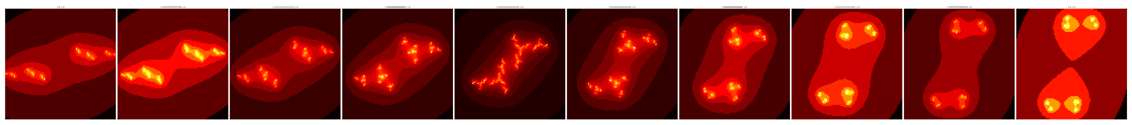
Finalmente,

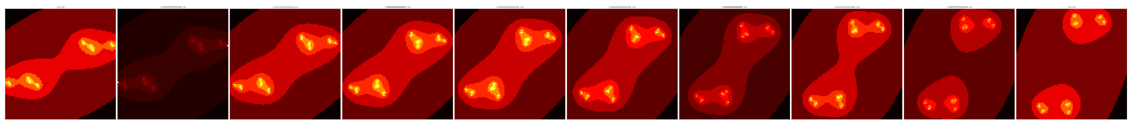
```
[ ]: # Plot every complex number in the julia set

     # 50 plots

     fig, axs = plt.subplots(5, 10, figsize=(100, 100))

     for i, x in enumerate(np.linspace(-2, 2, 10)):
             for j, y in enumerate(np.linspace(-2, 2, 5)):
                     axs[j, i].imshow(julia_set_fractal(complex(x, y)), cmap='hot')
                     axs[j, i].axis('off')
                     axs[j, i].set_title(f"{x}, {y}")

     fig.tight_layout()

     plt.show()
```











[ ]:

```

```