"Año del Fortalecimiento de la Soberania Nacional"

Universidad Nacional Mayor de San Marcos
(Universidad del Perú,DECANA DE AMERICA)

FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA Escuela Profesional de Ingenieria de Software

Segundo Laboratorio: Laboratorio Python

Curso: Programacion Paralela y Concurrente

Profesor: Edson Ticona Zegarra

Integrantes:

- Pichilingue Pimentel, Nathaly Nicole 19200247
- Torre Arteaga, Alexander 19200246
- Ricse Perez, Anthony Elias 19200276

Ciudad Universitaria 2022

1. **Explicación del algoritmo implementado**

   - **Cuestiones generales** Las distintas versiones del algoritmo realizado fueron implementadas en Python, haciendo uso del Python nativo, y liberías como Cython y OpenMP. Para realizar la evaluación de la entrada, se creó una función que genera una lista de n círculos con sus respectivos datos como posición xy, radio y color rgb. Luego, en base a esos datos, se traza el círculo, se detecta los píxeles afectados y se realiza la modificación en el archivo PPM, lo que permite dibujar los círculos deseados. Consideraciones a tomar en nuestro código:
     - Nuestro algoritmo solo toma como valores de entrada válidos, números enteros , negativos o positivos.
   - **Función principal** La función principal implementada en el proyecto llama a la función de entrada, donde ingresaremos los datos necesarios para dibujar el(los) circulo(s). Una vez se retorne la lista de cícrculos como respuesta, se llama a la función de creación de canvas en archivo PPM y luego a la función de dibujo, que nos retornará una matriz de píxeles. Con dicha matriz, se procede a llamar a la función de escritura de archivo, que modificará el canvas PPM creado incialmente según la posición y el canal de colores rgb de los píxeles obtenidos, finalizando así nla ejecución del programa.
   - **Función de entrada** La función de entrada llamada .$^{\text{en}}$tradaïnicia solicitando al usuario que ingrese el número de círculos a dibujar, una vez obtenido el número de cícrlos, se realiza una loop para solicitar los siguiente datos por cada círculo: coordenada (x,y) que representa el eje del círculo, radio del círculo, y valores para cada canal de color rgb del círculo, en el orden rojo, verde y azul. Luego, guarda dichos datos en una lista llamada círculos y retorna dicha lista.
   - **Función de creación de canvas** La función de creación del canvas, llamada create_image, genera una imagen del tamaño 1024 por 960 preestablecido multiplicando con el valor de los canales de color negro, representado en rgb como 000 (respectivamente para cada canal de color), y finalmente guarda el resultado en una lista, que es el retorno de la función.

- **Función de dibujo** La función de dibujo de círculos, llamada draw_circle, se ejecuta por cada n círculo a dibujar de la siguiente manera: Primero, realiza la lectura de los datos de cada círculo, asignando los valores en cada lugar de un array. Luego empieza a realizar iteraciones entre la coordenada x e y, en un rango de -radio a + radio, y procede a realizar dos verificaciones, la primera si el punto actual está fuera de los límites de la imagen, y la segunda si el punto actual está dentro del círculo, con la ecuación de la circunferencia

$$\sqrt{x} + \sqrt{y} = \sqrt{r} \tag{1}$$

  Una vez realizado dichas validaciones, guarda los datos del píxel según canal de color y una función XOR, y los inserta en una posición. Una vez generado la lista de píxeles con su posición y respectivo color rgb, la función retorna dicha lista de píxeles a modificar en la imagen.
- **Función de escritura de archivo** La función de escritura de archivo, llamada writePPM, genera la imagen de salida final, iniciando con el header del archivo, luego genera una lista llamada rgb donde almacenará el color de cada píxel según posición. una vez realizado esto se guarda la imagen en un "array.array" y escribe el archivo PPM, retornando como resultado el archivo de salida.

2. **Entorno de pruebas:**
   - Procesador: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz
   - Memoria RAM: 16 GB
   - Número de procesadores: 8
   - Sistema Operativo: Windows 10 Home (64 bits)
   - IDE de desarrollo y pruebas: Visual Studio Code
3. **Datos de entrada utilizados para las pruebas:**

- **Datos de entrada 1:**
  - Número de círculos: 10
  - Valores x, y, R, r, g, b por círculo:
    - Círculo 1: 649 -173 219 2 22 212
    - Círculo 2: 382 479 422 245 32 76
    - Círculo 3: 718 409 495 25 230 240
    - Círculo 4: -146 938 322 31 196 229
    - Círculo 5: 431 326 417 175 205 163
    - Círculo 6: 695 520 326 145 178 31
    - Círculo 7: -81 347 313 97 63 27
    - Círculo 8: -183 447 150 34 173 73
    - Círculo 9: 186 463 427 13 210 190
    - Círculo 10: 849 266 382 75 20 240
- **Datos de entrada 2:**
  - Número de círculos: 20
  - Valores x, y, R, r, g, b por círculo:
    - Círculo 1: -48 28 59 19 63 205
    - Círculo 2: 901 432 322 205 51 217
    - Círculo 3: 633 165 463 159 251 194
    - Círculo 4: -176 413 452 217 217 145
    - Círculo 5: 287 291 110 203 3 177
    - Círculo 6: 770 398 6 245 40 67
    - Círculo 7: 327 344 161 179 12 156
    - Círculo 8: 346 -73 19 8 90 155
    - Círculo 9: -10 -60 54 144 179 214
    - Círculo 10: -191 627 205 18 192 32
    - Círculo 11: -14 80 392 55 238 87
    - Círculo 12: 765 815 368 103 211 187
    - Círculo 13: 206 749 199 83 202 119
    - Círculo 14: 484 463 183 101 223 68
    - Círculo 15: 209 518 350 253 140 3
    - Círculo 16: 20 1004 217 227 192 181

- Círculo 17: 648 -189 136 157 28 140
- Círculo 18: 431 396 348 17 142 153
- Círculo 19: 202 160 387 196 122 171
- Círculo 20: 682 283 365 54 77 174

4. **Resultado de pruebas:** Poner texto aquí
5. **Análisis de resultados:** Poner texto aquí
6. **Anexo:**
    - Ray Tracer: https://github.com/aysusayin/Ray-Tracer